intra-mart WebPlatform/AppFramework Ver.7.0

アクセスセキュリティ仕様書

2013/10/18 第 9 版

<< 変更履歴 >>

変更年月日	変更内容
2008/07/07	初版
2008/08/29	第 2 版 アクセスセキュリティモジュールのインターフェースの下位互換対応（第 4 章）
2008/09/30	第 3 版 アクティブセッション管理を無効化するオプションの追加（第 5.4.4 節） エクスポートでのエンコーディングして機能追加（第 3.13.5 節）
2009/06/30	第 4 版 文章の訂正（第 5.1.2 節） SSORequestAnalyzer 誤植の訂正（第 4.7.2.3 節） ダイレクト URL の注意事項追加（第 5.1 節）
2009/10/30	第 5 版 日本語などマルチバイトを用いたプロパティファイルの変換方法を追記（第 2.42 節） ファイル保管場所の誤植の訂正（Servce -> Service）（第 3.14.1 節）
2010/11/30	第 6 版 リクエスト解析モジュール実装クラス「MobileAgentLoginRequestAnalyzer」を修正（第 4.7.2.3 節） jp.co.intra_mart.foundation.security.servlet.UserInitialServlet の説明を訂正（第 4.8.3 節） 既存アカウント属性定数（im_menu_folder_display_method）の定数値の誤りを訂正（第 3.5.3 節） スクリプト開発モデルのインポートサンプルの訂正（第 3.13.3 節） アクティブセッション情報操作実装モジュールにセッションID操作機能を追加(第 5.4.4.6 節) ［参考］ログイン中にログイン画面へのアクセスを防ぐ方法について追加(第 5.4.6 節)
2011/06/30	第 7 版 SSORequestAnalyzer の誤植を修正（第 4.7.2.3 節） アカウントの有効期間に関する注意を記述（第 3.5.1 節） インポートファイルのフォーマットに関する記述を追加（第 3.13 節）
2012/03/26	第 8 版 パスワード履歴管理操作モジュール実装クラスの処理フロー図追加（第 5.3.8 節）
2013/10/18	第 9 版 StandardAccountBackup で暗号化クラス名を指定できることを記載(第 3.13.5 節)

<< 目次 >>

1 アクセスセキュリティ概要 1
1.1 アクセスセキュリティ情報アーキテクチャ 1
1.1.1 アクセスセキュリティ情報操作アーキテクチャ 2
1.2 スクリプト開発でのデータベースのデータ取得アーキテクチャ 3
1.2.1 データベース変換アダプタインターフェース 4
1.2.2 データベース変換アダプタ実装クラス 4
1.3 アプリケーション情報 5
1.3.1 アプリケーション情報のインタフェース 5
1.4 ライセンス管理 6
2 システム（WebPlatform） 8
2.1 対応ロケール 9
2.1.1 対応ロケールへのアクセス 10
2.1.2 対応ロケール情報設定の定義 10
2.1.3 対応ロケール多言語情報 11
2.1.4 対応ロケール情報 11
2.1.5 ロケールのインタフェース 11
2.2 対応クライアント 12
2.2.1 対応クライアントへのアクセス 13
2.2.2 対応クライアントタイプ情報設定の定義 13
2.2.3 対応クライアント多言語情報 14
2.2.4 対応クライアント情報 14
2.2.5 クライアントタイプのインタフェース 14
2.3 対応カラーパターン 15
2.3.1 対応カラーパターンへのアクセス 16
2.3.2 カラーパターン情報設定の定義 16
2.3.3 対応カラーパターン多言語情報 17
2.3.4 カラーパターンプロパティ情報 17
2.3.5 対応カラーパターン情報 17
2.3.6 カラーパターンのインタフェース 17
2.4 メッセージ 18
2.4.1 メッセージ情報 18
2.4.2 メッセージの設定 18
2.4.3 メッセージへのアクセス 19
2.4.4 メッセージタグライブラリ 20
2.5 システムデータソース 21
2.5.1 システムデータソース情報 21
2.5.2 システムデータソースの設定 21
2.5.3 システムデータソースへのアクセス 22
2.6 パーミッションセレクタ 23
2.6.1 パーミッションセレクタの設定 23
2.6.2 パーミッションセレクタの呼出し 24
2.6.3 パーミッションセレクタの返却値について 24
2.6.4 パーミッションセレクタのインタフェース 24
2.6.5 パーミッションセレクタの実装クラス 25
2.7 システム管理者 27
2.7.1 システム管理者情報 27

2.7.2 システム管理者のモデル 27
2.7.3 システム管理者情報へのアクセス 28
2.7.4 システム管理者情報操作モジュール 28
2.7.5 システム管理者情報操作インタフェース 30
2.7.6 システム管理者情報操作実装クラス 30
2.8 システムメニュー 31
2.9 管理メニュー 32
2.9.1 管理メニュー情報 32
2.9.2 管理メニューのモデル 33
2.9.3 管理メニュー情報へのアクセス 33
2.9.4 管理メニュー情報操作モジュール 34
2.9.5 管理メニュー情報操作インタフェース 35
2.9.6 管理メニュー情報操作実装クラス 35
2.10 URLアクセスフィルタ 36
2.10.1 URLアクセスフィルタ設定の定義 36
2.10.2 URLアクセスフィルタのマッチング 37
3 ログイングループ 38
3.1 ログイングループ 38
3.1.1 ログイングループ情報 38
3.1.2 ログイングループのモデル 39
3.1.3 標準で用意されるメインページパターン 39
3.1.4 ログイングループ情報へのアクセス 40
3.1.5 ログイングループ情報操作モジュール 41
3.1.6 ログイングループ情報操作インタフェース 42
3.1.7 ログイングループ情報操作実装クラス 42
3.2 グループ管理者 43
3.2.1 グループ管理者情報 43
3.2.2 グループ管理者情報のモデル 43
3.2.3 グループ管理者情報へのアクセス 43
3.2.4 グループ管理者情報操作モジュール 44
3.2.5 グループ管理者情報操作インタフェース 45
3.2.6 グループ管理者情報操作実装クラス 45
3.3 グループデータソース 46
3.3.1 グループデータソース情報 46
3.3.2 グループデータソースの設定 46
3.3.3 グループデータソースへのアクセス 47
3.4 グループ管理権限ユーザメニュー 48
3.4.1 グループ管理権限ユーザメニュー情報へのアクセス 48
3.4.2 グループ管理権限ユーザメニュー情報操作モジュール 49
3.4.3 グループ管理権限ユーザメニュー情報操作インタフェース 51
3.4.4 アカウント情報操作実装クラス 51
3.5 アカウント 52
3.5.1 アカウント情報 52
3.5.2 アカウントのモデル 53
3.5.3 既存のアカウント属性 55
3.5.4 アカウント情報へのアクセス 56
3.5.5 アカウント情報操作モジュール 57

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.5.6 アカウント情報操作インタフェース....58
3.5.7 アカウント情報操作実装クラス....58
3.6 ロール....60
3.6.1 ロール情報....60
3.6.2 ロールのモデル....60
3.6.3 ロール情報へのアクセス....61
3.6.4 ロール情報操作モジュール....62
3.6.5 ロール情報操作インタフェース....63
3.6.6 ロール情報操作実装クラス....63
3.7 メニュー....64
3.7.1 メニュー情報....64
3.7.2 メニューのモデル....66
3.7.3 メニュー情報へのアクセス....66
3.7.4 メニュー情報操作モジュール....67
3.7.5 メニュー情報操作インタフェース....68
3.7.6 メニュー情報操作実装クラス....68
3.8 バッチ....69
3.8.1 バッチ情報....69
3.8.2 バッチのモデル....69
3.8.3 バッチ情報へのアクセス....70
3.8.4 バッチ情報操作モジュール....71
3.8.5 バッチ情報操作インタフェース....72
3.8.6 バッチ情報操作実装クラス....72
3.9 アクセスコントローラ....73
3.9.1 アクセスコントローラ情報....73
3.9.2 アクセスコントローラのモデル....74
3.9.3 アクセスコントローラ情報へのアクセス....74
3.9.4 アクセスコントローラ制御タグ....75
3.9.5 アクセスコントローラ情報操作モジュール....76
3.9.6 アクセスコントローラ情報操作インタフェース....77
3.9.7 アクセスコントローラ情報操作実装クラス....77
3.10 Webサービスアクセス....78
3.10.1 Webサービスアクセス情報....78
3.10.2 Webサービスアクセスのモデル....79
3.10.3 Webサービスアクセス情報へのアクセス....79
3.10.4 Webサービスアクセス情報操作モジュール....80
3.10.5 Webサービスアクセス情報操作インタフェース....82
3.10.6 Webサービスアクセス情報操作実装クラス....82
3.11 カレンダー....83
3.11.1 カレンダー表示情報....83
3.11.2 カレンダー表示情報のモデル....84
3.11.3 カレンダー表示情報へのアクセス....84
3.11.4 カレンダー表示情報操作モジュール....85
3.11.5 カレンダー表示情報操作インタフェース....86
3.11.6 カレンダー表示情報操作実装クラス....87
3.12 情報更新管理....89
3.12.1 情報更新管理マネージャ....89

3.12.2 既存の更新管理情報 ... 89
3.13 インポート／エクスポート ... 90
3.13.1 インポート／エクスポートのメソッド ... 90
3.13.2 インポート／エクスポートの設定 ... 91
3.13.3 インポート／エクスポートの呼出し ... 93
3.13.4 インポート／エクスポートのインタフェース ... 94
3.13.5 インポート／エクスポートの実装クラス ... 95
3.14 アクセスセキュリティの標準実装について ... 100
3.14.1 標準実装における情報の保管場所 ... 100
3.14.2 標準実装における注意点 ... 100
4 認証処理 ... 101
4.1 ログインユーザ ... 101
4.2 ログインシーケンス ... 102
4.2.1 ログインシーケンスの解説 ... 103
4.3 ログインセッション情報 ... 104
4.3.1 セッション情報インタフェース ... 104
4.3.2 ログインセッション情報設定の定義 ... 105
4.4 ログインリクエスト情報 ... 106
4.4.1 ログインリクエスト情報インタフェース ... 106
4.5 ログイン時のロケール ... 107
4.6 ログイン時のカラーパターン ... 108
4.7 認証処理の設定 ... 109
4.7.1 モジュール初期化インタフェース ... 109
4.7.2 リクエスト解析モジュール ... 110
4.7.3 認証モジュール ... 117
4.7.4 ページプロバイダモジュール ... 129
4.8 認証処理サーブレット ... 140
4.8.1 jp.co.intra_mart.foundation.security.servlet.SuperUserInitialServletクラス ... 140
4.8.2 jp.co.intra_mart.foundation.security.servlet.GroupSuperUserInitialServletクラス ... 140
4.8.3 jp.co.intra_mart.foundation.security.servlet.UserInitialServlet ... 141
4.8.4 jp.co.intra_mart.foundation.security.servlet.UserCertificationServlet ... 141
4.9 ログアウトサーブレット ... 142
4.9.1 jp.co.intra_mart.foundation.security.servlet.LogoutServletクラス ... 142
4.10 汎用サーブレット ... 143
4.10.1 jp.co.intra_mart.foundation.security.servlet.MenuServletクラス ... 143
4.11 クライアント別URLの設定 ... 144
4.11.1 URLの追加 ... 145
4.11.2 ページ作成 ... 146
4.12 ログイン画面の設定 ... 148
4.12.1 ユーザ名の 表示設定 ... 148
4.12.2 ログイン画面ヘッダ・フッタの指定 ... 148
4.13 タイトルバーの設定 ... 149
4.13.1 システム管理者 ... 149
4.13.2 グループ管理者／ 一般ユーザ（メインページパターン：MainPage510） ... 149
4.13.3 一般ユーザ（メインページパターン：システムデフォルト） ... 150
5 拡張機能 ... 151
5.1 ダイレクトURL ... 151

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.1.1 ダイレクトURLパラメータの作成 151
5.1.2 ダイレクトURLを使用する場合の注意点 151
5.2 ショートカットアクセス機能 152
5.2.1 ショートカットアクセスURL 152
5.2.2 ショートカットIDの作成 153
5.2.3 ショートカットアクセス機能を利用した場合の画面遷移 154
5.2.4 ショートカット拡張検証機能 154
5.2.5 ショートカットマネージャ 155
5.2.6 ショートカット情報操作アーキテクチャ 155
5.2.7 ショートカット情報操作設定の定義 155
5.2.8 ショートカット情報 156
5.2.9 ショートカット情報のモデル 156
5.2.10 ショートカット情報操作モジュールインタフェース 156
5.2.11 ショートカット情報操作実装モジュール 157
5.3 パスワード履歴管理機能 159
5.3.1 パスワード履歴管理機能における注意事項 159
5.3.2 パスワード履歴管理アーキテクチャ 159
5.3.3 パスワード履歴管理マネージャ 160
5.3.4 パスワード履歴管理情報設定の定義 160
5.3.5 パスワード履歴情報 161
5.3.6 パスワード履歴管理のモデル 161
5.3.7 パスワード履歴管理操作モジュールインタフェース 161
5.3.8 パスワード履歴管理操作モジュール実装クラス 162
5.4 二重ログイン防止機能 165
5.4.1 二重ログイン防止機能における注意事項 165
5.4.2 二重ログイン防止機能を利用した場合の画面遷移 165
5.4.3 アクティブセッション 166
5.4.4 アクティブセッションマネージャ 167
5.4.5 二重ログインマネージャ 173
5.4.6 [参考] ログイン中にログイン画面へのアクセスを防ぐ方法 177
5.5 暗号化モジュール 178
5.5.1 暗号化アルゴリズムの追加 178
5.6 汎用新着ポートレット連携 179
5.6.1 汎用新着ポートレット定義 179
5.6.2 パスワード有効期限通知の実装クラス 179
5.7 モバイル端末個体識別番号取得モジュール 180
5.7.1 モバイル端末個体識別番号取得メソッド 180
5.7.2 モバイル端末個体識別番号取得アーキテクチャ 180
5.7.3 モバイル端末個体識別番号取得設定ファイル 181
5.7.4 モバイル端末個体識別番号取得プロバイダインタフェース 181
5.7.5 モバイル端末個体識別番号取得モジュール実装クラス 181

1 アクセスセキュリティ概要

1.1 アクセスセキュリティ情報アーキテクチャ

ログイングループは、システムに複数存在し、各グループは独立したグループとして存在します。
システムにはすべての権限を保有するシステム管理者が存在します。
ログイングループ単位にグループ内のすべての権限を保有するグループ管理者が存在します。

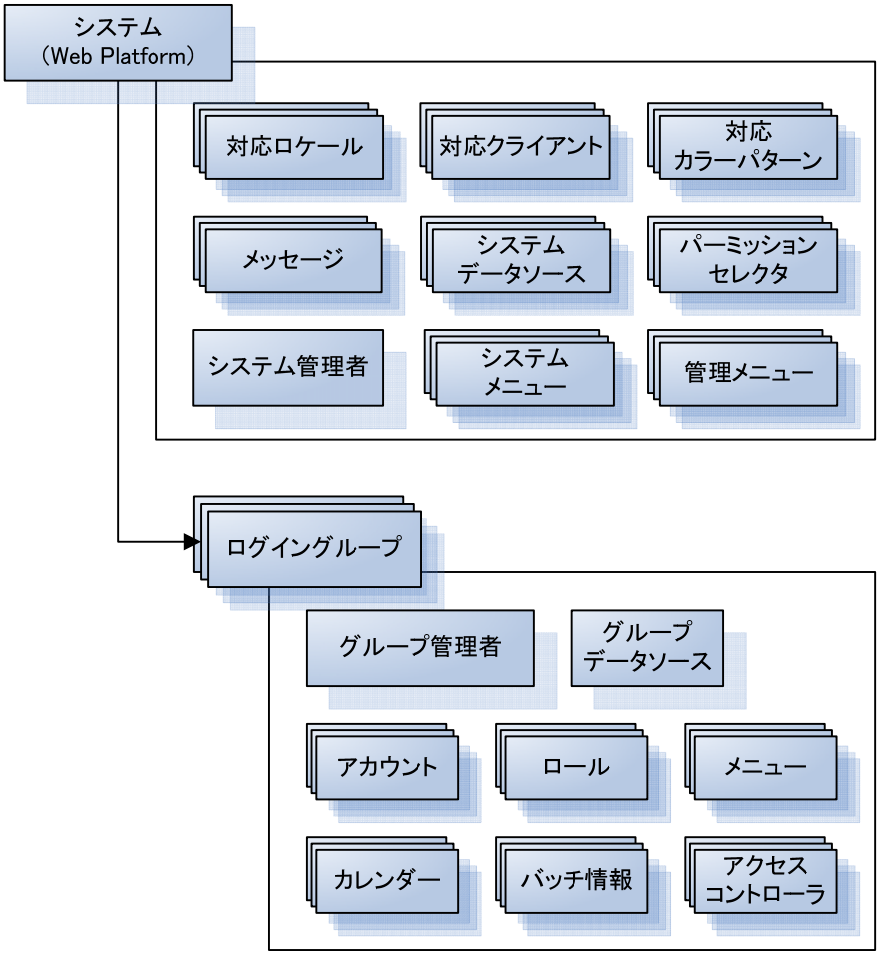


図 1 アクセスセキュリティ情報アーキテクチャ

1.1.1 アクセスセキュリティ情報操作アーキテクチャ

アクセスセキュリティで管理する各情報の操作は以下のアーキテクチャで構成されています。

- 参照系API、更新系APIにおいて、モジュールを呼び出す方式で実装されています。
- 更新系APIで利用するモジュールは、複数指定可能です。（複数のデータストアに保管可能）
- 各モジュール、リスナーは設定ファイル（サーバマネージャ/conf/access-security.xml）で設定します。<access-config>タグ内で設定します。
- 参照モジュールおよび更新モジュールは 1 つの実装クラスで定義し、設定ファイルに設定します。設定ファイルに設定されたモジュールの内、先頭のモジュールを参照モジュールとして利用します。
- 参照系APIでは、実際の参照モジュールの後に、修飾モジュールを呼び出すことが可能です。
- 更新系APIでは、実際の更新を行うモジュールの前後に、リスナーモジュールを呼び出すことが可能です。

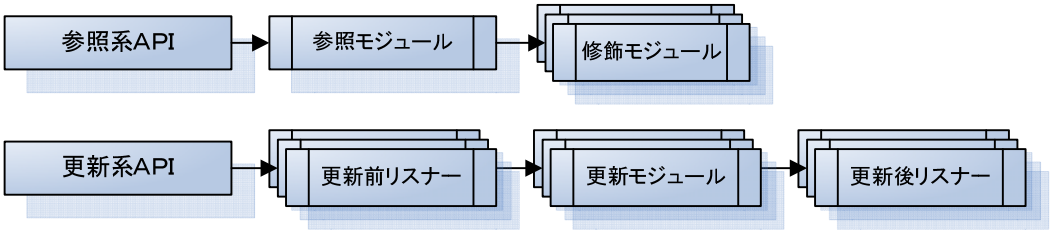


図 2 アクセスセキュリティ情報操作アーキテクチャ

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

1.2 スクリプト開発でのデータベースのデータ取得アーキテクチャ

スクリプト開発ではデータベースからのデータ取得時に、データベースの型から JavaScript の型への変換を自動的に行っています。
この変換処理は、データベースの型に応じて、データベース変換アダプタを用いて行っております。

データベース変換アダプタは、java.sql.Types のデータベース型毎に設定されています。
設定ファイルはサーバマネージャ/conf/database-adapter-config.xml です。

```
<?xml version="1.0" encoding="UTF-8"?>
<database-adapter-config>

  <database-adapter type="DATE" adapter-class="jp.co.intra_mart.foundation.database.DBResultDateAdapter"/>
  <database-adapter type="VARCHAR" adapter-class="jp.co.intra_mart.foundation.database.DBResultStringAdapter"/>
  ・・・・・・・

</database-adapter-config>
```

■ タグの説明

<database-adapter-config>			取得したシステム管理者情報を修飾するモジュールを設定します。 （複数指定可能）
	<database-adapter>		java.sql.Types のデータベース型対するデータベース変換アダプタを設定します。
			type 属性： java.sql.Types のデータベース型名を指定します。
			adapter-class 属性： データベース変換アダプタ実装クラスを指定します。 jp.co.intra_mart.foundation.database.DBResultAdapter インタフェースを実装する必要があります。
		その他のタグ	このクラスに渡す初期化パラメータです。(自由に設定可能) このクラスが生成された時に一度だけ<database-adapter>ノードが渡されて初期化します。

1.2.1 データベース変換アダプタインターフェース

■ **jp.co.intra_mart.foundation.database.DBResultAdapter インタフェース**

データベースから取得した情報を適切な JavaScript の型に変換するアダプタインタフェースです。

1.2.2 データベース変換アダプタ実装クラス

この実装クラスのパッケージ名は、jp.co.intra_mart.foundation.database です。

データベースの型 (java.sql.Types)	データベース変換アダプタ	説明
ARRAY	DBResultArrayAdapter	getArray()を利用して、Object の配列として値を返します。
BIGINT	DBResultBigIntAdapter	getLong()を利用して、 Number 型として値を返します。
BINARY	DBResultBinaryAdapter	getBytes()を利用して、エンコードしない String 型として取得します。
BIT	DBResultBooleanAdapter	getBoolean()を利用して、Boolean 型として値を返します。
BLOB	DBResultBlobAdapter	getBlob()を利用して、String 型として値を返します。
BOOLEAN	DBResultBooleanAdapter	getBoolean()を利用して、Boolean 型として値を返します。
CHAR	DBResultStringAdapter	getString() を利用して、文字列型として値を返します。
CLOB	DBResultClobAdapter	getClob()を利用して、String 型として値を返します。
DATALINK	DBResultObjectAdapter	getObject()を利用して Object 型として値を返します。
DATE	DBResultDateAdapter	getDate()を利用して、Date 型として値を返します。
DECIMAL	DBResultBigDecimalAdapter	getBigDecimal()を利用して、Number 型として値を返します。
DISTINCT	DBResultObjectAdapter	getObject()を利用して Object 型として値を返します。
DOUBLE	DBResultDoubleAdapter	getDouble()を利用して、Number 型として値を返します。
FLOAT	DBResultDoubleAdapter	getDouble()を利用して、Number 型として値を返します。
INTEGER	DBResultIntegerAdapter	getInt()を利用して、Number 型として値を返します。
JAVA_OBJECT	DBResultObjectAdapter	getObject()を利用して Object 型として値を返します。
LONGVARBINARY	DBResultBinaryAdapter	getBytes()を利用して、エンコードしない String 型として取得します。
LONGVARCHAR	DBResultStringAdapter	getString() を利用して、文字列型として値を返します。
NULL	DBResultNullAdapter	常に NULL を返します。
NUMERIC	DBResultBigDecimalAdapter	getBigDecimal()を利用して、Number 型として値を返します。
OTHER	DBResultObjectAdapter	getObject()を利用して Object 型として値を返します。
REAL	DBResultRealAdapter	getFloat()を利用して、Number 型として値を返します。
REF	DBResultRefAdapter	getRef()を利用して、Ref の参照を Object 型として値を返します。
SMALLINT	DBResultShortAdapter	getShort()を利用して、Number 型として値を返します。
STRUCT	DBResultStructAdapter	getStruct()を利用して、属性(java.sql.Struct.getAttributes())の Object 配列として値を返します。
TIME	DBResultTimeAdapter	getTime()を利用して、Date 型として値を返却します。
TIMESTAMP	DBResultImTimestampAdapter	imart.xml の intra-mart/platform/service/application/database/data/timestamp-is-date の値が true の場合は、getTimestamp()を利用して、Date 型として値を返します。 false の場合は、getString()を利用して、文字列型として値を返します。
TINYINT	DBResultByteAdapter	getByte()を利用して、Number 型として値を返します。
VARBINARY	DBResultBinaryAdapter	getBytes()を利用して、エンコードしない String 型として取得します。
VARCHAR	DBResultStringAdapter	getString() を利用して、文字列型として値を返します。
-------	DBResultTimestampAdapter	データベース TIMESTAMP 型の値を取得するアダプタです。 getTimeStamp()を利用して、Date 型として返却します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

1.3 アプリケーション情報

インストールされているアプリケーション情報およびシステム情報を保管するモデルです。

表 1 アプリケーション情報モデル(ApplicationInfo)

項目	説明	プロパティ（JavaScript）
アプリケーション ID	アプリケーション ID	**id**
アプリケーション名	アプリケーション名	**name**
アプリケーションタイプ	アプリケーションのタイプ	**type**
バージョン	バージョン	**version**
パッケージバーション	パッケージバーション	**packageVersion**
シリアル番号	シリアル番号	**serialNo**
最大サーバ数	最大サーバ数	**maxProcess**
最大ユーザ数	最大ユーザ数	**maxAccount**
試用版判定	試用版かどうか？	**trial**
フリーライセンス判定	フリーライセンスかどうか？	**free**
試用期限判定	試用期限がきているかどうか？	**expired**
使用期限付きライセンス判定	使用期限付きライセンスかどうか？	**limited**
ソースコード公開ライセンス判定	ソースコード公開ライセンスかどうか？	**openSource**
ランタイムライセンス判定	ランタイムライセンスかどうか？	**runtime**
インストール日付	アプリケーションがインストールされた日付	**installDate**
試用期限の日付	アプリケーションの試用期限の日付	**expiredDate**
期限付きライセンスの使用期限の日付	期限付きライセンスの使用期限の日付	**limitedDate**

1.3.1 アプリケーション情報のインタフェース

- **ApplicationInfo インタフェース**

  インストールされているアプリケーションの情報およびシステム情報を取得するインタフェース

1.4 ライセンス管理

ライセンス管理では、以下のライセンスについて管理を行います。

- システムライセンス情報の取得。
- ログイングループライセンスの取得、登録。
- アカウントライセンスの取得、登録、削除。
- アプリケーション情報の取得。
- アカウントに対するアプリケーションロールの取得、登録、削除。

ライセンス情報は、jp.co.intra_mart.foundation.security.license.LicenseManager を用いて取得および更新を行います。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2 システム（WebPlatform）

アクセスセキュリティにおいて、システム(WebPlatform)は全体を表します。
システムにおいては、以下の項目の管理を行います。

- 対応ロケール（多言語対応）の管理を行います。
- 対応クライアント（アクセスするクライアント）の管理を行います。
- 対応カラーパターン（表示画面色の系統パターン）の管理を行います。
- システム全体としてアクセスできるメッセージの管理を行います。
- システム全体を管理できるシステム管理者の管理を行います。
- システム管理者が利用するシステムメニューの管理を行います。
- システム全体としてアクセスできるデータソース（システムデータソース）の管理を行います。
- ログイングループ管理者が利用する管理メニューの管理を行います。
- ログイングループの管理を行います。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.1 対応ロケール

システムで扱うロケールの管理を行います。

- システム定義ファイルにシステムで利用できるロケールの一覧および各ロケールに対するデフォルトのエンコーディングを設定します。
- システム定義ファイルでは、システムでのデフォルトロケールを設定します。
- システムで利用できるロケールごとに表示用（ロケール表示名）の対応ロケール多言語定義ファイルを作成します。
- システム定義ファイルはサーバマネージャ/conf/system.xml です。
- 対応ロケール多言語定義ファイルはサーバマネージャ/conf/i18n に保管します。

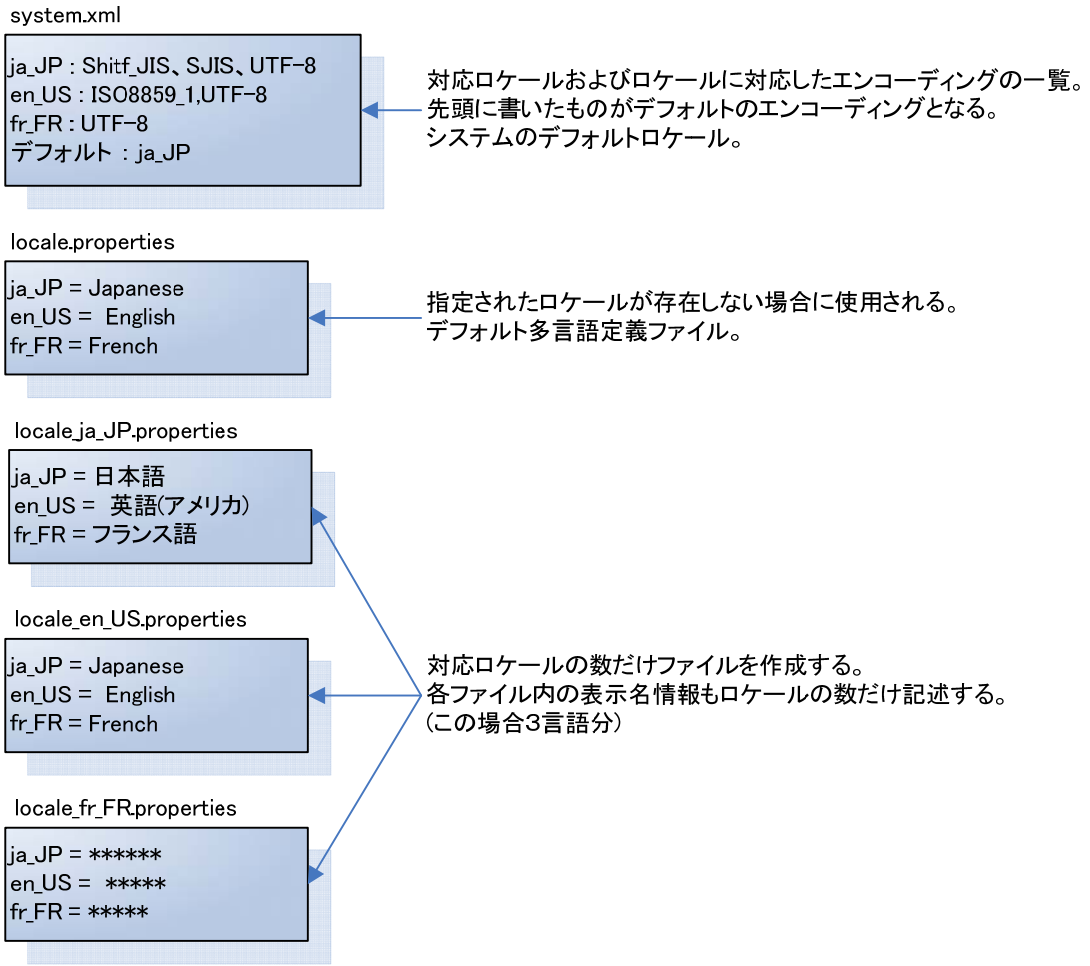


図 3 対応ロケール

2.1.1 対応ロケールへのアクセス

対応ロケールの取得は以下のように行います。

- スクリプト開発モデル
  System.getLocales のメソッドを利用します。

```
var locales = System.getLocales("ja");
```

上記の例では、日本語表示名で対応ロケール情報の一覧を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.SystemManager を利用します。

```
LocaleInfo[] locales = SystemManager.getInstance().getLocales(new Locale("ja"));
```

上記の例では、日本語表示名で対応ロケール情報の一覧を取得します。

2.1.2 対応ロケール情報設定の定義

- システムロケール情報設定ではシステムで取り扱うロケールの一覧を定義します。
- ロケール単位に対応するエンコーディング一覧も定義します。
- エンコーディング一覧で先頭に書いたものがデフォルトエンコーディングとなります。
- システムで取り扱うロケール内でデフォルトのロケールを定義します。
  デフォルトロケールは、ロケール一覧にあるものを設定してください。
- システム定義ファイル(サーバマネージャ/conf/system.xml)の locale-config タグ内で設定します。

locale-config での例を以下に示します。

```
<system>
  <locale-config>
    <locale>
      <locale-name>ja_JP</locale-name>
      <encoding-name>Shift_JIS</encoding-name>     デフォルト
      <encoding-name>UTF-8</encoding-name>
    </locale>
    <locale>
      <locale-name>en_US</locale-name>
      <encoding-name>8859-1</encoding-name>     デフォルト
      <encoding-name>UTF-8</encoding-name>
    </locale>
    <default-locale-name>ja_JP</default-locale-name>
  </locale-config>
</system>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.1.3 対応ロケール多言語情報

- ロケール別にロケールの表示名を定義します。
- システムで扱う対応ロケール多言語情報は、サーバマネージャ/conf/i18n に配置します。
- 対応ロケールに設定したロケールの種類分の多言語ファイルに加えてデフォルト多言語ファイルが必要です。

  多言語ファイルの種類 ＝ 対応ロケールの種類分の多言語ファイル ＋ デフォルト多言語ファイル

- ロケール別の多言語ファイル名は、locale_***.properties で作成します。（***はロケール文字列）
- デフォルト多言語ファイル名は、locale.properties で作成します。

```
ja_JP = Japanese
en_US = English
fr_FR = French
```

2.1.4 対応ロケール情報

表 2 対応ロケール情報モデル（LocaleInfo）

項目	説明	プロパティ（JavaScript）
ロケール表示名	ロケールの表示名	displayName
ロケール	ロケール	locale
デフォルトエンコーディング	このロケールのデフォルトエンコーディング	encoding
対応エンコーディング	システム管理者のエンコーディング	encodings

2.1.5 ロケールのインタフェース

- **LocaleProvider インタフェース**

  ロケールを取得するためのインタフェース。
  ロケール情報は必ずこのインタフェースを通して参照します。
  エンコーディング情報は必ずこのインタフェースを通して参照します。

- **LocaleInfo インタフェース**

  ロケールの情報を取得するためのインタフェース。
  ロケール情報は必ずこのインタフェースを通して参照します。

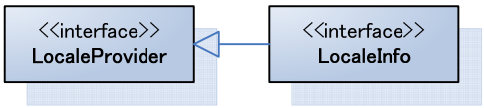


図 4 LocaleInfo インタフェース

2.2 対応クライアント

システムで利用可能なクライアントタイプの種類を管理します。

- システム定義ファイルにシステムで利用できるクライアントタイプの一覧を設定します。
  システム定義ファイルでは、システムでのデフォルトクライアントタイプを定義する。
- システムで利用できるロケールごとに表示用（クライアントタイプ表示名）の対応クライアント多言語定義ファイルを作成する。
- システム定義ファイルはサーバマネージャ/conf/system.xml です。
- 対応クライアント多言語定義ファイルはサーバマネージャ/conf/i18n に保管します。

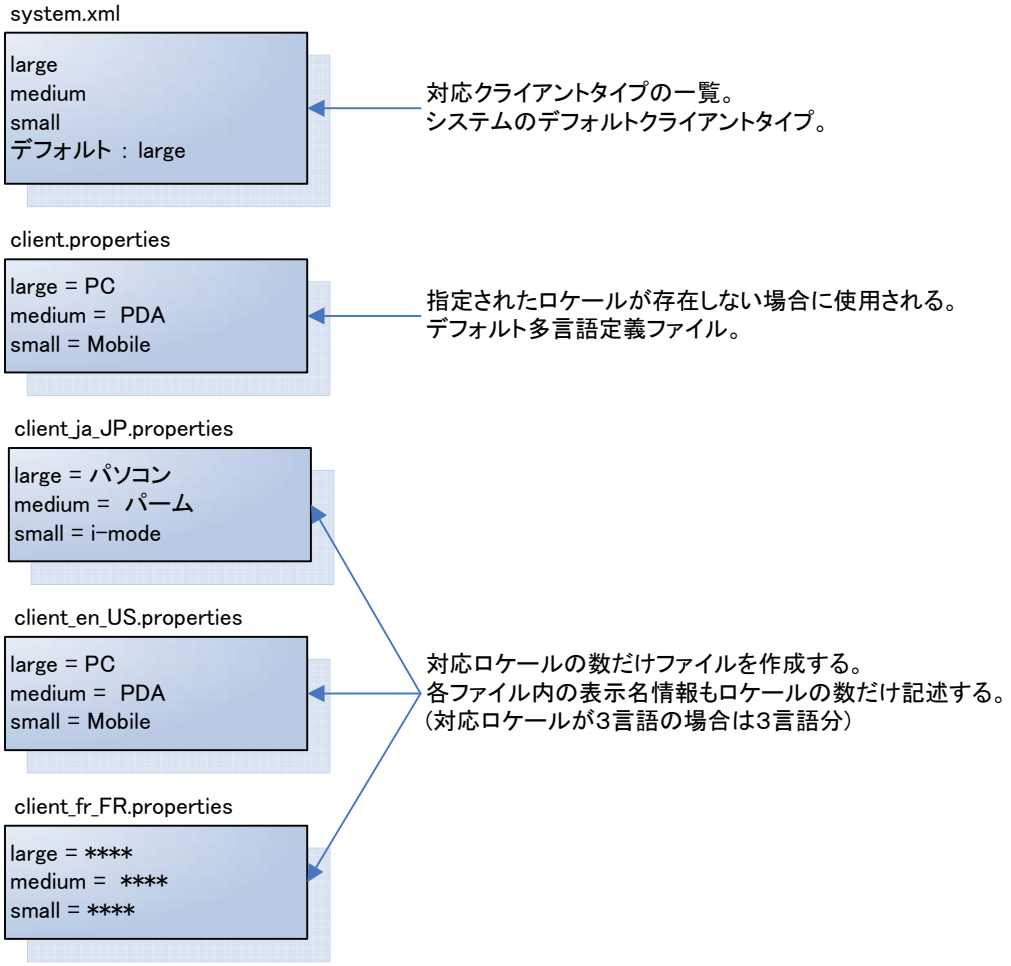


図 5 対応クライアント

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.2.1 対応クライアントへのアクセス

対応クライアントの取得は以下のように行います。

- スクリプト開発モデル
  System.getClientTypes のメソッドを利用します。

```
var clients = System.getClientTypes("ja");
```

上記の例では、日本語表示名で対応クライアント情報の一覧を取得します

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.SystemManager を利用します。

```
ClientTypeInfo[] clients = SystemManager.getInstance().getClientTypes(new Locale("ja"));
```

上記の例では、日本語表示名で対応クライアント情報の一覧を取得します。

2.2.2 対応クライアントタイプ情報設定の定義

- 対応クライアントタイプ情報設定ではシステムで取り扱うクライアントタイプの一覧を定義します。
- システムで取り扱うシステムクライアントタイプ内でデフォルトのクライアントタイプを定義します。
  デフォルトクライアントタイプは、クライアントタイプ一覧にあるものを設定してください。
- システム定義ファイル(サーバマネージャ/conf/system.xml)の client-config タグ内で設定します。

client-config での例を以下に示します。

```
<system>
  <client-config>
    <client-type-name>large</client-type-name>
    <client-type-name>medium</client-type-name>
    <client-type-name>small</client-type-name>
    <default-client-type-name>large</default-client-type-name>
  </client-config>
</system>
```

2.2.3 対応クライアント多言語情報

- ロケール別にクライアントタイプの表示名を定義します。
- システムで扱う対応クライアント多言語情報は、サーバマネージャ/conf/i18n に配置します。
- 対応クライアント多言語ファイルは対応ロケールの種類分の多言語ファイルに加えてデフォルト多言語ファイルが必要です。
  多言語ファイルの種類 ＝ 対応ロケールの種類分の多言語ファイル ＋ デフォルト多言語ファイル
- ロケール別の多言語ファイル名は、client_***.properties で作成します。(***はロケール文字列)
- デフォルト多言語ファイル名は、client.properties で作成します。

```
large = PC
medium = PDA
small = Mobile
```

2.2.4 対応クライアント情報

表 3 対応クライアント情報モデル(ClientTypeInfo)

項目	説明	プロパティ (JavaScript)
クライアントタイプ表示名	クライアントタイプの表示名	displayName
クライアントタイプ	クライアントタイプ	clientType

2.2.5 クライアントタイプのインタフェース

- **ClientTypeInfo インタフェース**
  クライアントタイプ情報を参照するインタフェース。
  システムのクライアントタイプ情報は必ずこのインタフェースを通して参照します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.3 対応カラーパターン

システムで利用可能なカラーパターンの種類を管理する。

- システム定義ファイルにシステムで利用できるカラーパターンの一覧を設定します。
- システム定義ファイルでは、システムでのデフォルトカラーパターンを定義します。
- システムで利用できるロケールごとに表示用（カラーパターン表示名）の対応カラーパターン多言語定義ファイルを作成する。
- システム定義ファイルはサーバマネージャ/conf/system.xml です。
- 対応クライアント多言語定義ファイルはサーバマネージャ/conf/i18n に保管します。

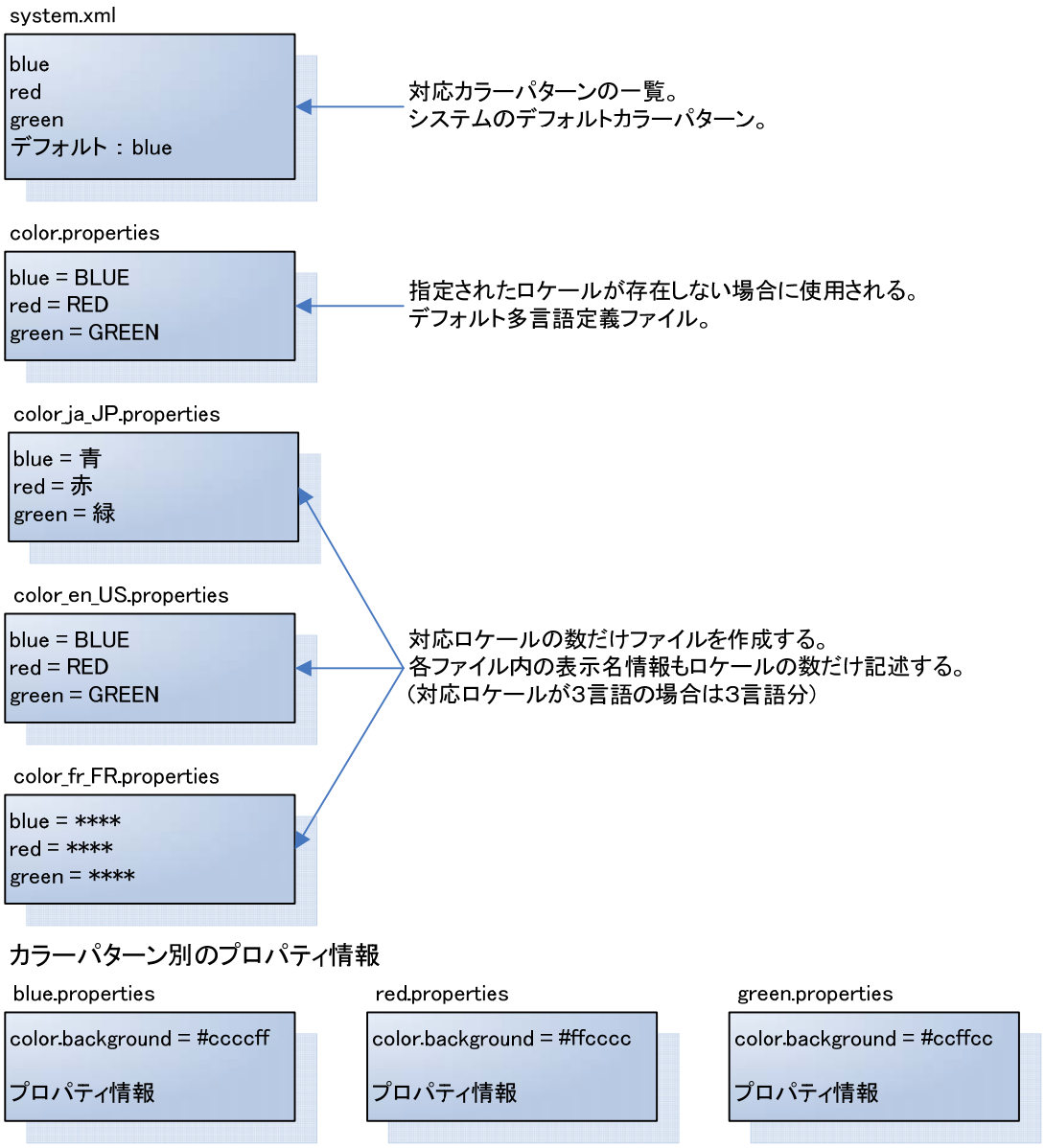


図 6 対応カラーパターン

2.3.1 対応カラーパターンへのアクセス

対応カラーパターンの取得は以下のように行います。

- スクリプト開発モデル
  System.getColorPatterns のメソッドを利用します。

```
var colors = System. getColorPatterns ("ja");
```

上記の例では、日本語表示名で対応カラーパターン情報の一覧を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.SystemManager を利用します。

```
ColorPatternInfo[] colors = SystemManager.getInstance().getColorPatterns(new Locale("ja"));
```

上記の例では、日本語表示名で対応カラーパターン情報の一覧を取得します。

2.3.2 カラーパターン情報設定の定義

- 対応カラーパターン情報設定ではシステムで取り扱うカラーパターン情報の一覧を定義します。
- システムで取り扱うシステムカラーパターン内でデフォルトカラーパターン ID を定義します。
  デフォルトカラーパターン ID は、カラーパターン一覧にあるものを設定してください。
- システム定義ファイル(サーバマネージャ/conf/system.xml)の color-config タグ内で設定します。

color-config での例を以下に示します。

```
<system>
  <color-config>
    <color-pattern>
        <color-pattern-id>blue</color-pattern-id>
        <color-code>#0000ff</color-code>
        <color-css-file>/skin/blue/blue.css</color-css-file>
        <color-css-file>/skin/blue/skin.css</color-css-file>
        <color-image-folder>/skin/bule</color-image-folder>
    </color-pattern>
    <color-pattern>
        <color-pattern-id>red</color-pattern-id>
        <color-code>#ff0000</color-code>
        <color-css-file>/skin/red/red.css</color-css-file>
        <color-css-file>/skin/red/skin.css</color-css-file>
        <color-image-folder>/skin/red</color-image-folder>
    </color-pattern>
    <default-color-pattern-id>blue</default-color-pattern-id>
  </color-config>
</system>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.3.3 対応カラーパターン多言語情報

- ロケール別にカラーパターンの表示名を定義します。
- システムで扱う対応カラーパターン多言語情報は、サーバマネージャ/conf/i18n に配置します。
- 対応カラーパターン多言語ファイルは対応ロケールの種類分の多言語ファイルに加えてデフォルト多言語ファイルが必要です。
  多言語ファイルの種類 ＝対応ロケールの種類分の多言語ファイル ＋ デフォルト多言語ファイル
- ロケール別の多言語ファイル名は、color_***.properties で作成する。（***はロケール文字列）
- デフォルト多言語ファイル名は、color.properties で作成する。

```
blue = BLUE
red = RED
```

2.3.4 カラーパターンプロパティ情報

- カラーパターンのプロパティ情報を定義します。
- システムで扱うカラーパターンプロパティ情報は、サーバマネージャ/conf/design に配置します。
- カラーパターンのプロパティ情報は,任意の情報を記述可能です。(キー=値形式)
- カラーパターンプロパティ情報ファイル名は、カラーパターン ID.properties で作成する。

2.3.5 対応カラーパターン情報

表 4 対応カラーパターン情報モデル（ColorPatternInfo）

項目	説明	プロパティ（JavaScript）
クライアントタイプ表示名	クライアントタイプの表示名	displayName
カラーパターン ID	カラーパターンの識別子	colorId
カラーコード	カラーパターンをイメージするカラーコード（#rrggbb 形式）	colorCode
CSS ファイルパス	CSS ファイルが格納されているパス。（コンテキストパスからの相対で、'/'から始める。）	cssFile
CSS ファイルパス(複数)	CSS ファイルが格納されているパスの一覧。（コンテキストパスからの相対で、'/'から始める。）	cssFiles
イメージフォルダパス	イメージが格納されているフォルダのパス。（コンテキストパスからの相対で、'/'から始める。）	imageFolder
カラーパターンプロパティ	カラーパターンプロパティ情報を取得する。	getProparty

2.3.6 カラーパターンのインタフェース

- **ColorPatternInfo インタフェース**
  カラーパターン情報を参照するインタフェース。
  カラーパターン情報は必ずこのインタフェースを通して参照します。

2.4 メッセージ

システム全体としてアクセスできるメッセージを管理します。

- メッセージはプロパティファイルに保管する方式をとっています。
- プロパティファイルは Java におけるプロパティファイルの仕様に準拠しています。
- プロパティファイルは対応言語ごとにサーバマネージャ/conf/message フォルダに保管します。
- サーバマネージャ/conf/message フォルダ内すべてのプロパティファイルの内容を結合して、起動時に読み込みを行います。（言語別）
- 同言語すべてのプロパティファイル内のメッセージ ID は、接頭語などを利用して、ユニークなものにしてください。（同じメッセージ ID があった場合は、後から読み込まれるファイルのものが有効になります。）

2.4.1 メッセージ情報

表 5 メッセージ情報モデル

項目	説明
メッセージ ID	メッセージID（入力必須）
ロケール	他言語対応用のロケール
メッセージ内容	メッセージの内容（入力必須）

2.4.2 メッセージの設定

- メッセージの設定は、プロパティファイルを対応言語ごとにサーバマネージャ/conf/message フォルダに保管します。
- プロパティファイル名は、サーバマネージャ/conf/message フォルダ内でユニークな名前である必要があります。
- Java のプロパティファイルの仕様に準じます。
  内容は、以下の通りです。
  メッセージ ID = メッセージ内容

- 日本語などマルチバイトを用いたプロパティファイルを作成した場合、Java コマンドである「naitive2asscii」を実行して、プロパティファイルを変換後にサーバに配置してください。

```
native2ascii マルチバイトを含んだプロパティファイルパス 変換後のプロパティファイルパス

native2ascii native2ascii_jp.properties native2ascii.properties
```

- メッセージ内容は、Javaの java.text.MessageFormat に準拠していますので、置換文字列が利用できます。{0}～{9}まで利用可能です。メッセージを取得する時に、引数で置換文字列を指定することで、メッセージに置換文字列が挿入されたものが取得されます。

```
msg1 = Success.
msg2 = {0} Not Found.
・・・・・・
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.4.3 メッセージへのアクセス

メッセージの取得は以下のように行います。

■ スクリプト開発モデル

MessageManager のメソッドを利用します。

```
var msg = MessageManager.getMessage("msg2","File");
```

上記の例では、"File Not Found."が取得されます。

■ JavaEE 開発モデル

jp.co.intra_mart.foundation.security.messageMessageManager を利用します。

```
String msg = MessageManager.getInstance().getMessage("msg2","File");
```

上記の例では、"File Not Found."が取得されます。

2.4.4 メッセージタグライブラリ

メッセージの取得には、タグライブラリも使用可能です。

■ スクリプト開発モデル

メッセージ ID と引数によりメッセージを生成します。

```
<IMART type="message" id="%message_id%" args="%args%" locale="%locale%">
```

属性名	名称	必須項目	説明
id	メッセージ ID	○	メッセージ ID を指定します。 この属性に設定できる値は、リテラルの文字列、String のいずれかです。
args	メッセージ引数		メッセージに差し込む値を設定します。 この属性に設定できる値は、 リテラルの文字列、String、String の Array のいずれかです。
locale	ロケール		取得するメッセージのロケールを設定します。 省略した場合は、ログインしているユーザのロケールを使用して、メッセージを取得します。 この属性に設定できる値は、リテラルの文字列、String のいずれかです。

■ J2EE 開発モデル

メッセージ ID と引数によりメッセージを生成します。

```
<imarttag::message id="%message_id%" args="%args%" locale="%locale%">
```

属性名	名称	必須項目	説明
id	メッセージ ID	○	メッセージ ID を指定します。 この属性に設定できる値は、リテラルの文字列、String のいずれかです。
args	メッセージ引数		メッセージに差し込む値を設定します。 この属性に設定できる値は、 リテラルの文字列、String Object[]　（各要素の toString()の値を使用します。） Collection　（各要素の toString()の値を使用します。） のいずれかです。
locale	ロケール		取得するメッセージのロケールを設定します。 省略した場合は、ログインしているユーザのロケールを使用して、メッセージを取得します。 この属性に設定できる値は、 リテラルの文字列、String、Locale のいずれかです。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.5 システムデータソース

システムデータソースは各ログイングループで共通にアクセスできるデータソースです。
接続 ID をユニークとして、複数設定することができます。
設定は、必須ではありません。必要に応じて設定してください。

2.5.1 システムデータソース情報

表 6 システムデータソース情報

項目	説明
接続 ID	システムデータソースの参照用の ID
データソース名	データソースの参照名

2.5.2 システムデータソースの設定

システムデータソースの設定は、サーバマネージャ/conf/data-source.xml ファイルを編集することで行います。

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

  <system-data-source>
    <connect-id>接続 ID</connect-id>
    <resource-ref-name>データソース参照名</resource-ref-name>
  </system-data-source>

</data-source>
```

■ 例 ： 接続1（connect1）と接続2（connect2）の設定を行います。

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

  <system-data-source>
    <connect-id>connect1</connect-id>
    <resource-ref-name>java:comp/env/jdbc/connect1</resource-ref-name>
  </system-data-source>

  <system-data-source>
    <connect-id>connect2</connect-id>
    <resource-ref-name>java:comp/env/jdbc/connect2</resource-ref-name>
  </system-data-source>

</data-source>
```

2.5.3 システムデータソースへのアクセス

システムデータソースへのアクセスは以下のように行います。

- スクリプト開発モデル

  DatabaseManager の各メソッドの引数に接続 ID を指定します。

```
var result = DatabaseManager.select(SQL,0,"connect1");
```

なお、最後の引数（接続 ID）を省略した場合は、ログインを行っているログイングループのデータソースに接続します。

- JavaEE 開発モデル

  jp.co.intra_mart.foundation.database.DatabaseManager を用いて、コネクションを取得します。

```
Connection con = DatabaseManager.getInstance().getSystemConnection("connect1");
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.6 パーミッションセレクタ

パーミッションセレクタは、絞込条件を元に、絞り込まれた情報を提供する機能です。
パーミッションセレクタは、1 つのカテゴリに属します。
1つのカテゴリには複数のパーミッションセレクタを含めることが可能です。
絞込みは 1 つのカテゴリを選択して行います。カテゴリに複数のパーミッションセレクタがある場合は、各パーミッションセレクタで絞り込まれた情報をマージして結果を返却します。
パーミッションセレクタの設定はサーバマネージャ/conf/access-security.xml で設定します。

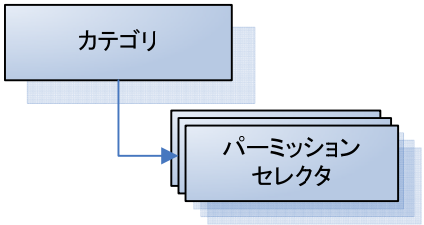


図 7 パーミッションセレクタの概要

2.6.1 パーミッションセレクタの設定

パーミッションセレクタは、設定ファイルに記述します。
設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <permission-config>

    <permission-group>
      <category>Category</category>
      <permission-selector>
        <permission-selector-class>PermissionSelectorClass</permission-selector-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </permission-selector>
    </permission-group>

  </permission-config>
</access-security>
```

■ タグの説明

<permission-group>		パーミッションセレクタをグループ化するタグです。（複数指定可能） このタグ内、カテゴリ名と複数のパーミッションセレクタを設定します。
	<category>	このパーミッショングループにカテゴリ名設定します。permission-config タグ内でユニークな ID を指定します。
	<permission-selector>	パーミッションセレクタの情報を設定します。（複数設定可能）
	<permission-selector-class>	パーミッションセレクタの実装クラスをを設定します。 PermissionSelector インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

2.6.2 パーミッションセレクタの呼出し

パーミッションセレクタの呼出しは以下ように行います。

- スクリプト開発モデル
  PermissionManager.doSelection メソッドを使用して絞込みを行います。

```
var manager = new PermissionManager("default");
var results = manager.doSelection("selector1","input");
```

上記の例では、ログイングループ ID（default）で, 'input'を絞込条件にカテゴリ'selector1'を指定して、絞込みを行います。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.permission.PermissionManager を用いて、絞込みを行います。

```
PermissionManager manager = new PermissionManager ("default");
Collection results = manager.doSelection("selector1","input");
```

上記の例では、ログイングループ ID（default）で, 'input'を絞込条件にカテゴリ'selector1'を指定して、絞込みを行います。

2.6.3 パーミッションセレクタの返却値について

JavaEE 開発モデルでは、オブジェクトの Collection 型で返却します。
スクリプト開発モデルでは、Collection 内の各オブジェクトを JavaBeans とみなし、そのプロパティを JavaScript オブジェクトのプロパティに変換して、Array 型で返却します。

2.6.4 パーミッションセレクタのインタフェース

パッケージは jp.co.intra_mart.foundation.security.permission になります。

- **PermissionSelector インタフェース**
  パーミッションセレクタの実装クラスに実装するインタフェース。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.6.5 パーミッションセレクタの実装クラス

■ **jp.co.intra_mart.foundation.security.permission.AdminMenuPermissionSelector クラス**

ログイングループを絞込条件にして、ログイングループ管理者に表示可能な管理メニューの一覧を返却するクラスです。

第二引数には、ログイングループ ID を指定します。

intra-mart WebPlatform 標準の実装クラスです。

カテゴリは **group-menu**

AdminMenuManager. getMenuTree メソッド内部での管理メニュー絞込みに使用しています。

図 8 AdminMenuPermissionSelector クラス

◆ 呼び出し方法

```
PermissionManager permissionManager = new PermissionManager(groupId);
Collection array = permissionManager.doSelection("group-menu", groupId);
String[] menuIds = (String[]) array.toArray(new String[array.size()]);
```

■ **jp.co.intra_mart.foundation.security.permission.MenuPermissionSelector クラス**

ユーザ情報を絞込条件にして、ユーザに表示可能なメニューの一覧を返却するクラスです。

第二引数には、ユーザ ID を指定します。

intra-mart WebPlatform 標準の実装クラスです。

カテゴリは **account-menu**

MenuManager. getAccountMenuTree メソッド内部でのメニュー絞込みに使用しています。

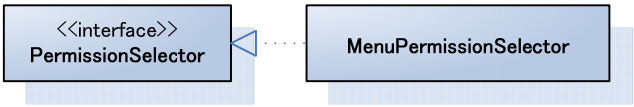


図 9 MenuPermissionSelector クラス

◆ 呼び出し方法

```
PermissionManager permissionManager = new PermissionManager(this.groupId);
String infos[] = new String[]{userId,strDate};
Collection array = permissionManager.doSelection("account-menu", infos);
String[] menuIds = (String[]) array.toArray(new String[array.size()]);
```

■ **jp.co.intra_mart.foundation.security.permission.AccountRolePermissionSelector クラス**

ユーザのアカウントロールと日付からからすべてのロール(サブロールも含む)を返却するクラスです。
現在の日付で取得する場合は、第二引数にユーザ ID を指定します。
指定日付で取得する場合は、第二引数は配列として指定し、{ユーザ ID、日付(文字列型)}を指定します。
日付(文字列型)のフォーマットは**'yyyy/mm/dd|hh:MM:ss'**です。
intra-mart WebPlatform 標準の実装クラスです。(Ver 6.1 以降)
カテゴリは **account-role**
AccountManager.getRoleIds メソッド内部でのロールの取得に使用しています。

図 10 AccountRolePermissionSelector クラス

◆ 呼び出し方法

```
PermissionManager permissionManager = new PermissionManager(this.groupId);
String infos[] = new String[]{userId,strDate};
Collection array = permissionManager.doSelection("account-role",infos);
String[] roles = (String[]) array.toArray(new String[array.size()]);
```

■ **jp.co.intra_mart.foundation.security.permission.AccessControllerPermissionSelector クラス**

ユーザ情報を利用して、有効なアクセスコントローラの一覧を返却するクラスです。
現在の日付で取得する場合は、第二引数にユーザ ID を指定します。
指定日付で取得する場合は、第二引数は配列として指定し、{ユーザ ID、日付(文字列型)}を指定します。
日付(文字列型)のフォーマットは'yyyy/mm/dd|hh:MM:ss'です。
intra-mart WebPlatform 標準の実装クラスです。(Ver 6.1 以降)
カテゴリは **access-controller**
AccessControllerManager.verify メソッド内部での権限チェックに使用しています。

図 11 AccessControllerPermissionSelector クラス

◆ 呼び出し方法

```
PermissionManager permissionManager = new PermissionManager(this.groupId);
String infos[] = new String[]{userId,strDate};
Collection array = permissionManager.doSelection("access-controller",infos);
String[] controllers = (String[]) array.toArray(new String[array.size()]);
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.7 システム管理者

システム管理者の権限は以下の通りです。

- すべてのログイングループの管理を行うことができます。
- ログイングループ管理者が利用する管理メニューの管理を行うことができます。

2.7.1 システム管理者情報

表 7 管理者モデル（SuperUser）

項目	説明	プロパティ（JavaScript）
ユーザ ID [必須]	システムが初回に起動された時に設定を行う。（デフォルトは system）	userId
パスワード	システム管理者のパスワード。（デフォルトは manager）	password
ロケール	システム管理者のロケール	locale
エンコーディング	システム管理者のエンコーディング	encoding
メールアドレス	システム管理者に対しての問い合わせを行うためのメールアドレス。	mailAddrress
電話番号	システム管理者に対しての問い合わせを行うための電話番号。	telNo

2.7.2 システム管理者のモデル

- **jp.co.intra_mart.foundation.security.admin.SuperUser クラス**

  管理者情報を提供するクラスです。

  - システム管理者、グループ管理者で共通に利用します。
  - 管理者情報は必ずこのクラスを通して操作します。

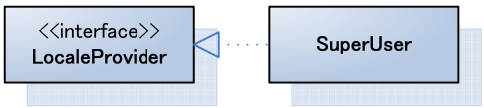


図 12 SuperUser クラス

2.7.3 システム管理者情報へのアクセス

システム管理者情報の取得は以下のように行います。

■ スクリプト開発モデル

System.getSuperUser メソッドを使用して取得します。

```
var su = System.getSuperUser();
```

■ JavaEE 開発モデル

jp.co.intra_mart.foundation.security.SystemManager を用いて、システム管理者情報を取得します。

```
SuperUser su = SystemManager.getInstance().getSuperUser();
```

2.7.4 システム管理者情報操作モジュール

実際にシステム管理者情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <super>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>PermanentSuperUserAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </super>
  </access-config>
</access-security>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ タグ説明

<accessor>	システム管理者情報を取得、更新するモジュールを設定します。（複数指定可能）	
	<accessor-class>	システム管理者情報を取得するクラスを設定します。 SuperUserAccessor インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>	取得したシステム管理者情報を修飾するモジュールを設定します。 （複数指定可能）	
	<reader-decorator-class>	取得したシステム管理者情報を修飾するクラスを設定します。 SuperUserReaderDecorator インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>	システム管理者情報を更新した後に呼ばれるモジュールを設定します。 （複数指定可能）	
	<writer-listener-class>	システム管理者情報を更新する前に呼ばれるクラスを設定します。 SuperUserQueryWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>	システム管理者情報を更新した後に呼ばれるモジュールを設定します。 （複数指定可能）	
	<writer-listener-class>	システム管理者情報を更新した後に呼ばれるクラスを設定します。 SuperUserPostWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

2.7.5 システム管理者情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.admin になります。

- **SuperUserAccessor インタフェース**
  システム管理者情報を取得、更新するモジュールのインタフェース。

- **SuperUserReaderDecorator インタフェース**
  取得したシステム管理者情報を修飾するモジュールのインタフェース。

- **SuperUserQueryWriterListener インタフェース**
  システム管理者情報を更新する前に呼ばれるモジュールのインタフェース。

- **SuperUserPostWriterListener インタフェース**
  システム管理者情報を更新した後に呼ばれるモジュールのインタフェース。

2.7.6 システム管理者情報操作実装クラス

- **jp.co.intra_mart.system.security.PermanentSuperUserAccessor クラス**
  PermanentDataService に対してシステム管理者情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。（非公開のクラスとなります。）

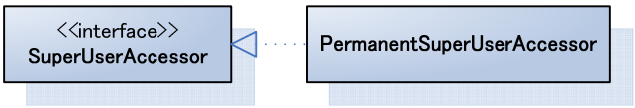


図 13 PermanentSuperUserAccessor クラス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.8 システムメニュー

システムメニューは、システム管理者がログインを行った場合に利用するメニューを管理します。

- システムメニューの設定は、設定ファイル（サーバマネージャ/conf/system-menu.xml）で設定します。
- メニューの表示名は対応しているロケールの数だけ、設定する必要があります。
- メニューの URL はスクリプト開発モデルの URL のみが指定できます。
- 設定ファイルは必ず UTF-8 で保存してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<system-menu>
  <menu>
    <display-name>
      <locale>ja</locale>
      <locale-display-name>システム環境</locale-display-name>
    </display-name>
    <display-name>
      <locale>en</locale>
      <locale-display-name>System Status</locale-display-name>
    </display-name>
    <url>system/security/system/main_frame.jssp</url>
  </menu>
<menu>
  <display-name>
    <locale>ja</locale>
    <locale-display-name>システム管理者設定</locale-display-name>
  </display-name>
  <display-name>
    <locale>en</locale>
    <locale-display-name>System Administrator Setting</locale-display-name>
  </display-name>
  <url>system/setting/system/super_user/main.jssp</url>
</menu>

....

</system-menu>
```

2.9 管理メニュー

管理メニューは、グループ管理者が使用するメニューです。
すべてのログイングループのグループ管理者で共通のメニューを使用します。

2.9.1 管理メニュー情報

管理メニューは階層構造となります。
管理メニューには、フォルダとメニュー項目の2種類があります。

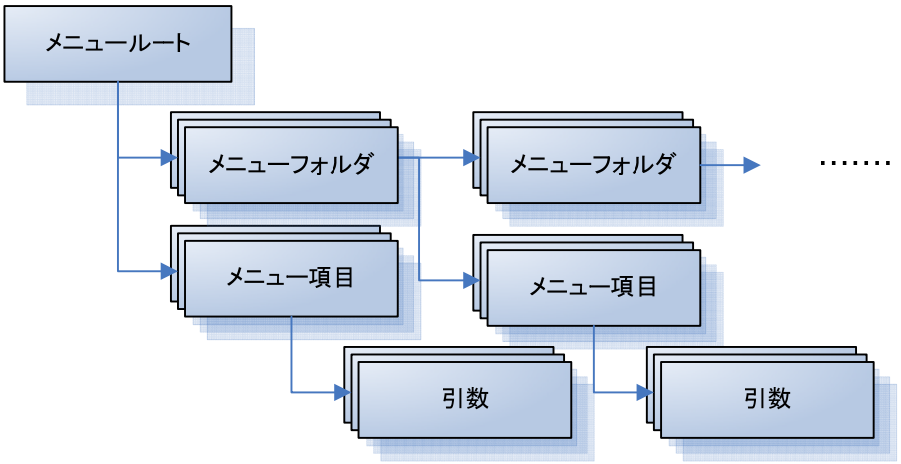


図 14 管理メニュー構成図

表 8 管理メニュー情報モデル

項目	説明	メニューフォルダ	メニュー項目	プロパティ (JavaScript)
メニューID	メニューID	○	○	menuId
メニュー名	メニューの表示名	○	○	displayName
ノードタイプ	メニューフォルダかメニュー項目かを指定	○	○	type
説明	メニューの説明	○	○	description
ソート番号	ソートするための番号			sortNumber
URL	表示するURL		○	url
引数	引き渡すパラメータ		○	argument
権限フラグ	表示権限を有効にするかどうかのフラグ 無効の場合は、すべてのグループ管理者のメニューに表示されます。	○	○	accessControlEnabled

*○は必須項目です。

■ メニュー親子関係

メニューの親子関係を表すモデルです。

表 9 メニュー親子関係情報モデル

項目	説明	プロパティ (JavaScript)
親メニューID	親メニューID	**parentMenuId**
子メニューID	子メニューID	**childMenuId**

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ メニュー引数
メニュー内容を表示する際に、リクエストに引き渡す引数です。
メニュー項目のみ有効です。

表 10 メニュー引数情報モデル

項目	説明
引数名	引数のキー名
引数値	引数の値

■ 管理メニュー権限
メニューを表示するための権限です。（複数設定可能）
ログイングループ毎に管理メニューの表示非表示を行うために用います。
管理メニュー権限に設定されたログイングループのグループ管理者のメニューに表示されます。
管理メニュー情報モデルの権限フラグが有効の場合のみこの機能が働きます。

表 11 管理メニュー権限 ログイングループ

項目	説明
ログイングループ ID	ログイングループ ID

2.9.2 管理メニューのモデル

■ **jp.co.intra_mart.foundation.security.menu.MenuItem クラス**
メニュー情報を提供するクラスです。
システムメニュー、管理メニュー、メニューで共通に利用します。
メニュー情報は必ずこのクラスを通して操作します。

■ **jp.co.intra_mart.foundation.security.menu.MenuInclusion クラス**
メニュー情報の親子関係を提供するクラスです。
システムメニュー、管理メニュー、メニューで共通に利用します。
メニュー情報の親子関係は必ずこのクラスを通して操作します。

2.9.3 管理メニュー情報へのアクセス

管理メニュー情報の取得は以下のように行います。

■ スクリプト開発モデル
AdminMenuManager.getMenuItem メソッドを使用して取得します。

```
var menu = AdminMenuManager.getMenuItem("menuId");
```

■ JavaEE 開発モデル
jp.co.intra_mart.foundation.security.menu.AdminMenuManager を用いて、管理メニュー情報を取得します。

```
MenuItem menu = AdminMenuManager.getInstance().getMenuItem ("menuId");
```

2.9.4 管理メニュー情報操作モジュール

実際に管理メニュー情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <admin-menu>

      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>

      <accessor>
        <accessor-class>PermanentAdminMenuAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>

      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>

      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </admin-menu>

  </access-config>
</access-security>
```

■ タグ説明

<accessor>	管理メニュー情報を取得、更新するモジュールを設定します。(複数指定可能)	
	<accessor-class>	管理メニュー情報を取得するクラスを設定します。 AdminMenuAccessor インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>	取得した管理メニュー情報を修飾するモジュールを設定します。(複数指定可能)	
	<reader-decorator-class>	取得した管理メニュー情報を修飾するクラスを設定します。 AdminMenuReaderDecorator インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

<query-writer-listener>	管理メニュー情報を更新する前に呼ばれるモジュールを設定します。 （複数指定可能）	
	<writer-listener-class>	管理メニュー情報を更新する前に呼ばれるクラスを設定します。 AdminMenuQueryWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>	管理メニュー情報を更新した後に呼ばれるモジュールを設定します。 （複数指定可能）	
	<writer-listener-class>	管理メニュー情報を更新した後に呼ばれるクラスを設定します。 AdminMenuPostWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

2.9.5 管理メニュー情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.menu になります。

- **AdminMenuAccessor インタフェース**
  管理メニュー情報を取得、更新するモジュールのインタフェース。

- **AdminMenuReaderDecorator インタフェース**
  取得した管理メニュー情報を修飾するモジュールのインタフェース。

- **AdminMenuQueryWriterListener インタフェース**
  管理メニュー情報を更新する前に呼ばれるモジュールのインタフェース。

- **AdminMenuPostWriterListener インタフェース**
  管理メニュー情報を更新した後に呼ばれるモジュールのインタフェース。

2.9.6 管理メニュー情報操作実装クラス

- **jp.co.intra_mart.system.security.PermanentAdminMenuAccessor クラス**
  PermanentDataService に対して管理メニュー情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。（非公開のクラスとなります。）

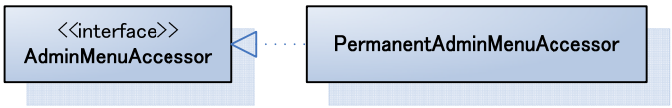


図 15 PermanentAdminMenuAccessor クラス

2.10 URLアクセスフィルタ

URL アクセスフィルタでは、URL 指定によるアクセスを制御することができます。

- ログインしたユーザの種類（システム管理者、グループ管理者、ユーザ）によって、URL アクセスを制御可能です。
- URL アクセス制御は、URL マッピングによって行います。
- URL マッピングは正規表現で指定することも可能です。
- URL マッピングは設定ファイル（サーバマネージャ/conf/access-security.xml）で行います。

2.10.1 URLアクセスフィルタ設定の定義

システム管理者、アカウント、ユーザ別にアクセスできる URL のマッピングを設定できます。

<url-pattern>では、簡単な URL パターンを指定できます。
例 /group/* ・・・ /group/以下のすべての URL とマッチする。
*.jsp ・・・ 拡張子が jsp である URL とマッチする。

<regexp-pattern>では、正規表現を用いた URL パターンを指定できます。
例 /group.*/.* ・・・ /group を含む URL とマッチする。

filter-config での例を以下に示します。

```
<access-security>
  <filter-config>

    <system-filter>
      <url-pattern>super/*</url-pattern>
      <regexp-pattern>*.jsp</regexp-pattern>
      ・・・
    </sytem-filter>

    <login-group-filter>
      <url-pattern>group/*</url-pattern>
      ・・・
    </login-group-filter>

    <user-filter>
      <url-pattern>user/*</url-pattern>
     ・・・
    </user-filter>

  </filter-config>
</access-security>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

2.10.2 URLアクセスフィルタのマッチング

URL アクセスフィルタのマッチングは、以下のルールで行います。

アクセスできない URL の場合は、アクセス件なしのエラー画面を出力します。

表 12 URL アクセスフィルタの一覧

アクセス状態	URL のパス	アクセス
ログインしていない。	<system-filter>のパターンに一致	×
	<group-filter のパターンに一致	×
	<user-filter のパターンに一致	×
	それ以外のパス	○
システム管理者でログイン	<system-filter>のパターンに一致	○
	<group-filter のパターンに一致	○
	<user-filter のパターンに一致	○
	それ以外のパス	○
グループ管理者でログイン	<system-filter>のパターンに一致	×
	<group-filter のパターンに一致	○
	<user-filter のパターンに一致	○
	それ以外のパス	○
ユーザでログイン	<system-filter>のパターンに一致	×
	<group-filter のパターンに一致	×
	<user-filter のパターンに一致	○
	それ以外のパス	○

3 ログイングループ

3.1 ログイングループ

この章では、ログイングループ単位で管理を行う情報について解説します。

3.1.1 ログイングループ情報

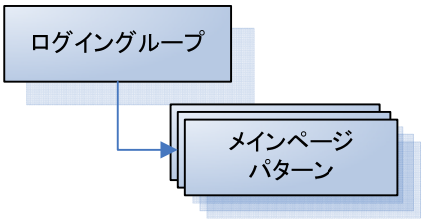


図 16 ログイングループ情報

表 13 ログイングループ情報モデル

項目	説明	プロパティ (JavaScript)
ログイングループID [必須]	ログイングループ ID	id
ロケール	ログイングループでのデフォルトロケール	locale
エンコーディング	ログイングループでのデフォルトエンコーディング	encoding
アカウントロック回数	この回数だけ連続してログインに失敗したら アカウントをロックする。(正の整数) 0 の場合は、アカウントロックは行わない。	lockCount
アカウントロック期間	アカウントがロックされてからこの期間が経過した場合は アカウントロックを解除する。(単位は分) 0 の場合は、アカウントロックの解除は行わない。	lockTerm

■ メインページパターン情報

メインページパターン情報はログイングループ毎に複数設定できます。

◆ ログイングループのユーザは、このメインページパターンの中からログイン後のページを選択することが出来ます。

表 14 ログイングループ メインページパターン情報モデル

項目	説明	プロパティ (JavaScript)
ページパターン ID [必須]	ページパターン ID	id
ページ URL [必須]	ページの URL	url
ページイメージ URL	ページイメージの URL(画像ファイル)	imageUrl

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.1.2 ログイングループのモデル

■ **jp.co.intra_mart.foundation.security.group.LoginGroup クラス**

ログイングループ情報を提供するクラスです。
ログイングループ情報は必ずこのクラスを通して操作します。

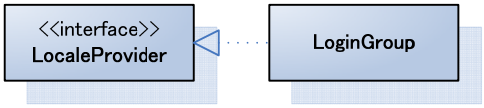


図 17 LoginGroup クラス

■ **jp.co.intra_mart.foundation.security.group.MainPagePatternInfo クラス**

メインページパターン情報を提供するクラスです。
メインページパターン情報は必ずこのクラスを通して操作します。

3.1.3 標準で用意されるメインページパターン

ログイングループ作成時に 1 種類のメインページパターンが設定されます。
ユーザはデフォルトのメインページとあわせて、2 種類のメインページを選択することができます。

表 15 標準のメインページパターン

パターン ID	説明
デフォルト	ページプロバイダが返却するメインページを表示します。
MaimPage510	Ver.5.1 のメインページデザイン

ログイングループ作成時に追加されるメインページパターンは、
StorageService の storage/system/basic/init-main-page.xml
に記述されています。
このファイルを編集することで、ログイングループ作成時に追加されるメインページパターンを変更することができます。

3.1.4 ログイングループ情報へのアクセス

ログイングループ情報の取得は以下のように行います。

- スクリプト開発モデル
  LoginGroupManager.getLoginGroup メソッドを使用して取得します。

```
var loginGroup = LoginGroupManager.getLoginGroup("default");
```

上記の例では、ログイングループ ID（default）のログイングループ情報を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.group.LoginGroupManager を用いて、ログイングループ情報を取得します。

```
LoginGroup loginGroup = LoginGroupManager.getInstance().getLoginGroup("default");
```

上記の例では、ログイングループ ID（default）のログイングループ情報を取得します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.1.5 ログイングループ情報操作モジュール

実際にログイングループ情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <group>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>PermanentLoginGroupAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </group>
  </access-config>
</access-security>
```

■ タグ説明

<accessor>	ログイングループ情報を取得、更新するモジュールを設定します。 （複数指定可能）	
	<accessor-class>	ログイングループ情報を取得するクラスを設定します。 LoginGroupAccessor インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>	取得したログイングループ情報を修飾するモジュールを設定します。 （複数指定可能）	
	<reader-decorator-class>	取得したログイングループ情報を修飾するクラスを設定します。 LoginGroupReaderDecorator インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>	ログイングループ情報を更新する前に呼ばれるモジュールを設定します。 （複数指定可能）	
	<writer-listener-class>	ログイングループ情報を更新する前に呼ばれるクラスを設定します。 LoginGroupQueryWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>	ログイングループ情報を更新した後に呼ばれるモジュールを設定します。 （複数指定可能）	
	<writer-listener-class>	ログイングループ情報を更新した後に呼ばれるクラスを設定します。 LoginGroupPostWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

3.1.6 ログイングループ情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.group になります。

- **LoginGroupAccessor インタフェース**
  ログイングループ情報を取得、更新するモジュールのインタフェース。

- **LoginGroupReaderDecorator インタフェース**
  取得したログイングループ情報を修飾するモジュールのインタフェース。

- **LoginGroupQueryWriterListener インタフェース**
  ログイングループ情報を更新する前に呼ばれるモジュールのインタフェース。

- **LoginGroupPostWriterListener インタフェース**
  ログイングループ情報を更新した後に呼ばれるモジュールのインタフェース。

3.1.7 ログイングループ情報操作実装クラス

- **jp.co.intra_mart.system.security.PermanentLoginGroupAccessor クラス**
  PermanentDataService に対してログイングループ情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。（非公開のクラスとなります。）

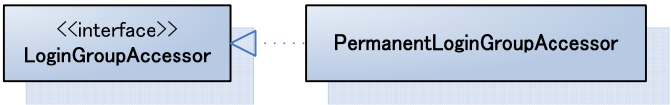


図 18 PermanentLoginGroupAccessor クラス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.2 グループ管理者

3.2.1 グループ管理者情報

表 16 グループ管理者情報モデル

項目	説明	プロパティ (JavaScript)
ユーザ ID [必須]	システムが初回に起動された時に設定を行う。(デフォルトは system)	userId
パスワード	システム管理者のパスワード。(デフォルトは manager)	password
ロケール	システム管理者のロケール	locale
エンコーディング	システム管理者のエンコーディング	encoding
メールアドレス	システム管理者に対しての問い合わせを行うためのメールアドレス。	mailAddrress
電話番号	システム管理者に対しての問い合わせを行うための電話番号。	telNo

3.2.2 グループ管理者情報のモデル

- **jp.co.intra_mart.foundation.security.admin.SuperUser クラス**
  管理者情報を提供するクラスです。
  システム管理者、グループ管理者で共通に利用します。
  管理者情報は必ずこのクラスを通して操作します。

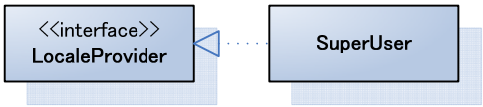


図 19 SuperUser クラス

3.2.3 グループ管理者情報へのアクセス

グループ管理者情報の取得は以下のように行います。

- スクリプト開発モデル
  LoginGroupManager.getSuperUser メソッドを使用して取得します。

```
var su = LoginGroupManager.getSuperUser("default");
```

上記の例では、ログイングループ ID(default)のグループ管理者情報を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.group.LoginGroupManager を用いて、ログイングループ情報を取得します。

```
SuperUser su = LoginGroupManager.getInstance().getSuperUser("default");
```

上記の例では、ログイングループ ID(default)のグループ管理者情報を取得します。

3.2.4 グループ管理者情報操作モジュール

実際にグループ管理者情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <group-super>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>PermanentGroupSuperUserAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </group-super>
  </access-config>
</access-security>
```

■ タグ説明

<accessor>	グループ管理者情報を取得、更新するモジュールを設定します。(複数指定可能)	
	<accessor-class>	グループ管理者情報を取得するクラスを設定します。 GroupSuperUserAccessor インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>	取得したグループ管理者情報を修飾するモジュールを設定します。 (複数指定可能)	
	<reader-decorator-class>	取得したグループ管理者情報を修飾するクラスを設定します。 GroupSuperUserReaderDecorator インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>	グループ管理者情報を更新する前に呼ばれるモジュールを設定します。 (複数指定可能)	
	<writer-listener-class>	グループ管理者情報を更新する前に呼ばれるクラスを設定します。 GroupSuperUserQueryWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>	グループ管理者情報を更新した後に呼ばれるモジュールを設定します。 (複数指定可能)	
	<writer-listener-class>	グループ管理者情報を更新した後に呼ばれるクラスを設定します。 GroupSuperUserPostWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.2.5 グループ管理者情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.group になります。

- **GroupSuperUserAccessor インタフェース**
  グループ管理者情報を取得、更新するモジュールのインタフェース。

- **GroupSuperUserReaderDecorator インタフェース**
  取得したグループ管理者情報を修飾するモジュールのインタフェース。

- **GroupSuperUserQueryWriterListener インタフェース**
  グループ管理者情報を更新する前に呼ばれるモジュールのインタフェース。

- **GroupSuperUserPostWriterListener インタフェース**
  グループ管理者情報を更新した後に呼ばれるモジュールのインタフェース。

3.2.6 グループ管理者情報操作実装クラス

- **jp.co.intra_mart.system.security.PermanentGroupSuperUserAccessor クラス**
  PermanentDataService に対してグループ管理者情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。（非公開のクラスとなります。）

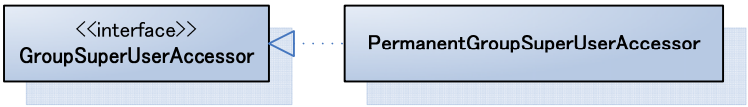


図 20 PermanentGroupSuperUserAccessor クラス

3.3 グループデータソース

データソースはログイングループ毎に 1 つのみ設定できるデータソースです。
設定は、ログイングループ毎に1つだけ必ず必要です。

3.3.1 グループデータソース情報

表 17 グループデータソース情報モデル

項目	説明
ログイングループ ID	ログイングループ ID
データソース名	データソースの参照名

3.3.2 グループデータソースの設定

グループデータソースの設定は、サーバマネージャ/conf/data-source.xml ファイルを編集することで行います。

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

  <group-data-source>
    <login-group-id>ログイングループ ID</login-group-id>
    <resource-ref-name>データソース参照名</resource-ref-name>
  </group-data-source>

  ……

</data-source>
```

例 ： ログイングループ 1（group1）とログイングループ2（group2）の設定を行います。

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

  <group-data-source>
    <login-group-id>group1</login-group-id>
    <resource-ref-name>java:comp/env/jdbc/group1</resource-ref-name>
  </group-data-source>

  <group-data-source>
    <login-group-id>group2</login-group-id>
    <resource-ref-name>java:comp/env/jdbc/group1</resource-ref-name>
  </group-data-source>

……

</data-source>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.3.3 グループデータソースへのアクセス

ログイングループのデータソースへのアクセスは以下のように行います。

■ スクリプト開発モデル

DatabaseManager の各メソッドの引数にグループ ID を指定します。

```
var result = DatabaseManager.select(SQL);
```

または

```
var result = DatabaseManager.select(SQL,0,"group1",true);
```

■ JavaEE 開発モデル

jp.co.intra_mart.foundation.database.DatabaseManager を用いて、コネクションを取得します。

```
Connection con = DatabaseManager.getInstance().getLoginGroupConnection("group1");
```

3.4 グループ管理権限ユーザメニュー

グループ管理者が利用できる管理メニューを一般ユーザに委譲したときの管理メニュー情報を管理します。

ユーザがグループ管理者としてログインできるかどうか(グループ管理権限オプション)は、アカウントの属性に保管します。
属性名については「3.5.3 既存のアカウント属性」を参照してください。

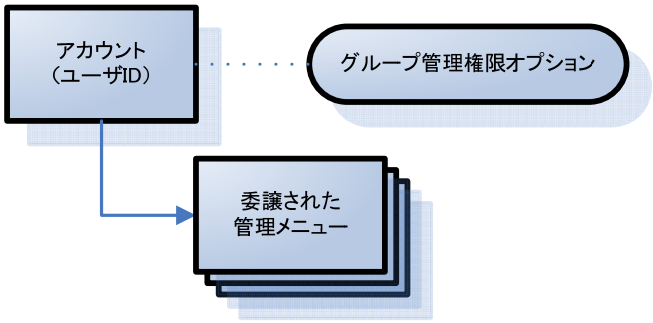


図 21 グループ管理権限ユーザメニューアーキテクチャ

3.4.1 グループ管理権限ユーザメニュー情報へのアクセス

ユーザに権限が委譲された管理メニューID 一覧の取得は以下のように行います。

- スクリプト開発モデル
  AdminUserManager.getMenuIdsByUserId()メソッドを使用して取得します。

```
var manager = new AdminUserManager("default");
var menuIds = manager.getMenuIdsByUserId ("user1");
```

上記の例では、ログイングループ ID(default),ユーザ ID(user1)に委譲された管理メニューID 一覧を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.adminuser.AdminUserManager.getMenuIdsByUserId()メソッドを用いて、管理メニューID 一覧を取得します。

```
AdminUserManager manager = new AdminUserManager("default");
String[] menuIds = manager.getMenuIdsByUserId("user1");
```

上記の例では、ログイングループ ID(default),ユーザ ID(user1)に委譲された管理メニューID 一覧を取得します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.4.2 グループ管理権限ユーザメニュー情報操作モジュール

実際にアカウント情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/admin-user-config.xml です。

```
<admin-user-config>
  <access-config>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>StandardAccountAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
  </access-config>

  <backup-config>
    <backup-group>
      <category>standard</category>
      <import>
        <import-class>SampleInpoter</import-class>
      </import>
      <export>
        <export-class>SampleExporter</export-class>
      </export>
    </backup-group>
  </backup-config>
</admin-user-config>
```

■ タグ説明

<access-config>			グループ管理権限ユーザメニュー情報操作機能の設定です。
	<accessor>		グループ管理権限ユーザメニュー情報を取得するモジュールの設定します。
		<acsessor-class>	グループ管理権限ユーザメニュー情報を、更新するモジュールの実装クラスを設定します。(複数指定可能) jp.co.intra_mart.foundation.security.adminuser.AdminUserAccessor を実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。
	<reader-decorator>		取得したグループ管理権限ユーザメニュー情報を修飾するモジュールを設定します。(複数指定可能)
		<reader-decorator-class>	取得したグループ管理権限ユーザメニュー情報を修飾するクラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserReaderDecorator インタフェースを実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。
	<query-writer-listener>		グループ管理権限ユーザメニュー情報を更新する前に呼ばれるモジュールを設定します。(複数指定可能)
		<writer-listener-class>	グループ管理権限ユーザメニュー情報を更新する前に呼ばれるクラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserQueryWriterListener インタフェースを実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。
	<post-writer-listener>		グループ管理権限ユーザメニュー情報を更新した後に呼ばれるモジュールを設定します。(複数指定可能)
		<writer-listener-class>	グループ管理権限ユーザメニュー情報を更新した後に呼ばれるクラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserPostWriterListener インタフェースを実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<backup-config>				グループ管理権限ユーザメニュー情報のインポート/エクスポート機能の設定です。
	<backup-group>			インポート/エクスポートのグループ設定です。(複数設定可能)
		<category>		インポート/エクスポートのグループの名称です。 <backup-config>内でユニークな値です。
		<import>		インポートに関する設定です。
			<import-class>	グループ管理権限ユーザメニュー情報のインポート実装クラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserImporter を実装する必要があります。
			<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。
		<export>		エクスポートに関する設定です。
			<export-class>	グループ管理権限ユーザメニュー情報のエクスポート実装クラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserExporterAccessor を実装する必要があります。
			<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.4.3 グループ管理権限ユーザメニュー情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.adminuser になります。

- **AdminUserAccessor インタフェース**
  グループ管理権限ユーザメニュー情報を取得、更新するモジュールのインタフェース。

- **AdminUserReaderDecorator インタフェース**
  取得したグループ管理権限ユーザメニュー情報を修飾するモジュールのインタフェース。

- **AdminUserQueryWriterListener インタフェース**
  グループ管理権限ユーザメニュー情報を更新する前に呼ばれるモジュールのインタフェース。

- **AdminUserPostWriterListener インタフェース**
  グループ管理権限ユーザメニュー情報を更新した後に呼ばれるモジュールのインタフェース。

3.4.4 アカウント情報操作実装クラス

- **jp.co.intra_mart.foundation.security.adminuser.StandardAdminUserAccessor クラス**
  ログイングループの標準データソースに対してグループ管理権限ユーザメニュー情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。

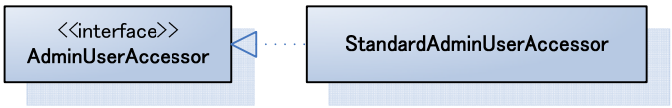


図 22 StandardAdminUserAccessor クラス

3.5 アカウント

3.5.1 アカウント情報

アカウント情報はログイングループに属します。(複数設定可能)

アカウント情報のユーザ ID はログイングループ内でユニークな ID にする必要があります。

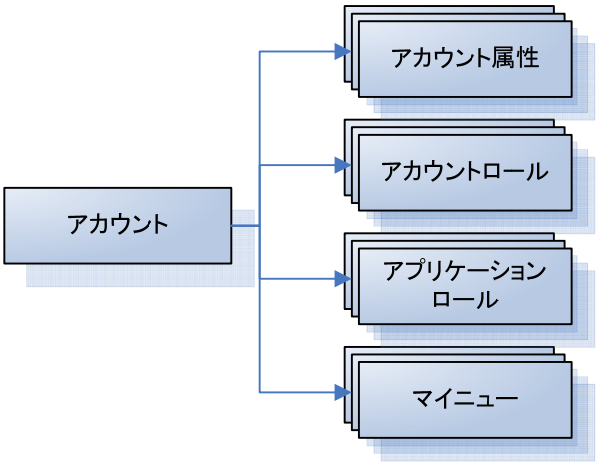


表 18 アカウント情報モデル

項目	説明	プロパティ (JavaScript)
ユーザ ID [必須]	ログインアカウント(入力必須)	userId
ロケール	アカウントのデフォルトロケール	locale
エンコーディング	ログイングループでのデフォルトエンコーディング	encoding
パスワード	ログインパスワード	password
モバイル用パスワード	モバイル用パスワード	mobilePassword
説明	アカウントの説明。	note
有効開始日	アカウントの有効開始日を設定する。	validStartDate
有効終了日	アカウントの有効終了日を設定する	validEndDate
外出フラグ	外出しているかどうかのフラグ	goOut
アカウントロック日付	アカウントがロックされた場合に、日付を設定する。	lockDate
ログイン失敗回数	ログイン失敗の回数を保管する。 ログインできた場合は、0 にクリアする。	loginFailureCount
メインページパターン ID	メインページパターン ID を保管する。	mainPagePatternId
カラーパターン ID	各種ページのカラーパターンを保管する	colorPatternId

(注意) 有効期間判定は有効期間開始日 ≦ 日時 < 有効期間終了日となっています。

以下のような範囲で設定したい場合

2010/09/01 - 2010/09/20

有効開始日および有効終了日には

2010/09/01 00:00:00 - 2010/09/21 00:00:00

を指定する必要があります。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ アカウントロール情報

アカウントロール情報はアカウント毎に複数設定できます。

表 19 アカウント アカウントロール情報モデル

項目	説明	プロパティ (JavaScript)
ロールID	ロールID	roleId
有効開始日	アカウントロールの有効開始日を設定する。	validStartDate
有効終了日	アカウントロールの有効終了日を設定する。	validEndDate

■ マイメニュー情報

マイメニュー情報はアカウント毎に複数設定できます。

表 20 アカウント マイメニュー情報モデル

項目	説明	プロパティ (JavaScript)
メニューID	メニューID	menuId
表示名	マイメニューの表示名	name
イメージパス	マイメニューのアイコンイメージ	imagePath
ソート番号	マイメニューをソートする番号	sortNum

■ アカウント属性情報

アカウント属性情報はアカウント毎に複数設定できます。

属性キー名、属性値には文字列のみ使用できます。

表 21 アカウント アカウント属性情報モデル

項目	説明
属性キー名	属性のキー名です。
属性値	属性の値です。

■ アプリケーションロール情報

アプリケーションロール情報はアカウント毎に複数設定できます。

表 22 アカウント アプリケーションロール情報モデル

項目	説明
アプリケーション ID	アプリケーションID

3.5.2 アカウントのモデル

■ **jp.co.intra_mart.foundation.security.account.Account クラス**

アカウント情報を提供するクラスです。

アカウント情報は必ずこのクラスを通して操作します。

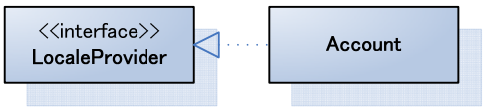


図 23 Account クラス

- **jp.co.intra_mart.foundation.security.account.AccountRole クラス**
  アカウントロール情報を提供するクラスです。
  アカウントロール情報は必ずこのクラスを通して操作します。

- **jp.co.intra_mart.foundation.security.account.FavoriteMenu クラス**
  マイメニュー情報を提供するクラスです。
  マイメニュー情報は必ずこのクラスを通して操作します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.5.3 既存のアカウント属性

アカウント属性には、既存の属性が存在します。この属性はシステムで利用しています。

スクリプト開発モデルでは、AccountManager の定数として定義されています。

JAVAEE 開発モデルでは、jp.co.intra_mart.foundation.security.accountt.AccountManager の定数として宣言されています。

表 23 既存のユーザ属性

属性	説明	属性定数名および値	
メニュー 表示・非表示設定 （Ver.6.0）	メインページの初期表示において、左側のメニューを表示非表示するかどうかの設定です。（“true” / “false”）	定数名	ATTR_MENU_DISPLAY
		定数値	im_menu_display
マイメニュー 表示位置設定 （Ver.6.0）	マイメニューの表示位置を上部か左側のメニュー内に表示するかの設定です。 トップバー ： “IM_TOP” 左側メニュー内 ： “IM_MENU”	定数名	ATTR_MY_MENU_POSITION
		定数値	im_my_menu_position
パスワード期限切れ 通知設定 （Ver.6.0）	パスワード履歴管理において、パスワードの期限切れを何日前から汎用新着ポートレットに表示するかの設定です。（日数）	定数名	ATTR_REPORT_LIMIT_DAYS
		定数値	im_report_limit_days
メニューフォルダの 表示・非表示設定 （Ver.6.1）	メニューのフォルダ名をクリックした時に、メイン画面にそのフォルダのメニュー項目を一覧表示するかどうかの設定です。 （“standard” / “IM_MAIN”）	定数名	ATTR_MENU_FOLDER_DISPLAY_METHOD
		定数値	im_menu_folder_display_method
モバイル端末 個体識別番号設定 （Ver.6.1）	ユーザのモバイル端末個体識別番号を保管する属性です。	定数名	ATTR_MOBILE_EASY_LOGIN
		定数値	im_mobile_easy_login
初回ログイン設定 （仮パスワード） （Ver.7.0）	初回ログインかどうかを設定します。 （“true” / “false”）	定数名	ATTR_FIRST_LOGIN
		定数値	im_first_login
グループ管理権限 ユーザ設定 （Ver.7.0）	グループ管理権限ユーザかどうかを設定します。 （“true” / “false”）	定数名	ATTR_ADMIN_USER_OPTION
		定数値	im_admin_option

3.5.4 アカウント情報へのアクセス

アカウント情報の取得は以下のように行います。

- スクリプト開発モデル

  AccountManager.getAccount メソッドを使用して取得します。

```
var manager = new AccountManager("default");
var account = manager.getAccount("guest");
```

上記の例では、ログイングループ ID（default）,ユーザ ID（guest）のアカウント情報を取得します。

- JavaEE 開発モデル

  jp.co.intra_mart.foundation.security.account.AccountManager を用いて、アカウント情報を取得します。

```
AccountManager manager = new AccountManager("default");
Account account = manager.getAccount("guest");
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.5.5 アカウント情報操作モジュール

実際にアカウント情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <account>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>StandardAccountAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </account>
  </access-config>
</access-security>
```

■ タグ説明

<accessor>		アカウント情報を取得、更新するモジュールを設定します。 （複数指定可能）
	<accessor-class>	アカウント情報を取得するクラスを設定します。 AccountAccessor インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>		取得したアカウント情報を修飾するモジュールを設定します。 （複数指定可能）
	<reader-decorator-class>	取得したアカウント情報を修飾するクラスを設定します。 AccountReaderDecorator インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>		アカウント情報を更新する前に呼ばれるモジュールを設定します。 （複数指定可能）
	<writer-listener-class>	アカウント情報を更新する前に呼ばれるクラスを設定します。 AccountQueryWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>		アカウント情報を更新した後に呼ばれるモジュールを設定します。 （複数指定可能）
	<writer-listener-class>	アカウント情報を更新した後に呼ばれるクラスを設定します。 AccountPostWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

3.5.6 アカウント情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.account になります。

- **AccountAccessor インタフェース**
  アカウント情報を取得、更新するモジュールのインタフェース。

- **AccountReaderDecorator インタフェース**
  取得したアカウント情報を修飾するモジュールのインタフェース。

- **AccountQueryWriterListener インタフェース**
  アカウント情報を更新する前に呼ばれるモジュールのインタフェース。

- **AccountPostWriterListener インタフェース**
  アカウント情報を更新した後に呼ばれるモジュールのインタフェース。

3.5.7 アカウント情報操作実装クラス

- **jp.co.intra_mart.foundation.security.StandardAccountAccessor クラス**
  ログイングループの標準データソースに対してアカウント情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。

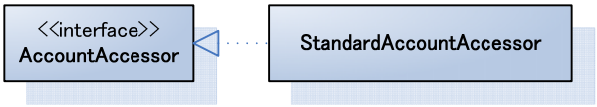


図 24 StandardAccountAccessor クラス

◆ このクラスはパラメータをとることが可能です。

パラメータ名	パラメータ値
cryption-class	パスワード、およびモバイルパスワードを暗号化して保存するためのクラスを指定します。 設定するクラスは、jp.co.intra_mart.foundation.security.cryption.Cryption インタフェースを実装する必要があります。 省略した場合は、intra-mart WebPlatform 標準の暗号化アルゴリズムで暗号化します。 標準の暗号化アルゴリズムの実装クラスは jp.co.intra_mart.foundation.security.cryption.StandardCryption です。 また、暗号化を行わない実装クラスとして jp.co.intra_mart.foundation.security.cryption.NoCryption が用意されています。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

◆ 設定例

```
<access-security>
  <access-config>
    <account>
      ・・・・
      <accessor>
        <accessor-class> jp.co.intra_mart.foundation.security.StandardAccountAccessor</accessor-class>
        <init-param>
          <param-name>cryption-class</param-name>
          <param-value>暗号化クラス名</param-value>
        </init-param>
      </accessor>
      ・・・・
    </account>
  </access-config>
</access-security>
```

3.6 ロール

3.6.1 ロール情報

ロール情報はログイングループに属します。(複数設定可能)

ロール情報のロール ID、ロール名はログイングループ内でユニークな値にする必要があります。

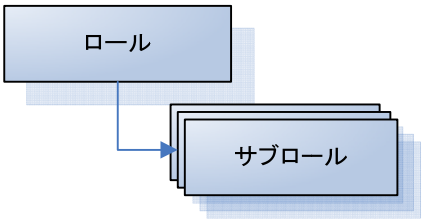


表 24 ロール情報モデル

項目	説明	プロパティ (JavaScript)
ロール ID [必須]	ロールID	roleId
ロール名 [必須]	ロール名	roleName
表示名 [必須]	ロールの表示名	**displayName**
説明	ロールの説明	**description**
カテゴリ	ロールのカテゴリ	category

■ サブロール情報

サブロール情報はロール毎に複数設定できます。

表 25 ロール サブロール情報モデル

項目	説明
ロール ID	サブロールID

3.6.2 ロールのモデル

■ **jp.co.intra_mart.foundation.security.role.Role クラス**

ロール情報を提供するクラスです。

ロール情報は必ずこのクラスを通して操作します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.6.3 ロール情報へのアクセス

ロール情報の取得は以下のように行います。

■ スクリプト開発モデル

RoleManager.getRole メソッドを使用して取得します。

```
var manager = new RoleManager("default");
var role = manager.getRole("guest");
```

上記の例では、ログイングループ ID（default）,ロール ID（guest）のロール情報を取得します。

■ JavaEE 開発モデル

jp.co.intra_mart.foundation.security.Role.RoleManager を用いて、ロール情報を取得します。

```
RoleManager manager = new RoleManager("default");
Role role = manager.getRole("guest");
```

上記の例では、ログイングループ ID（default）,ロール ID（guest）のロール情報を取得します。

3.6.4 ロール情報操作モジュール

実際にロール情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <role>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>StandardRoleAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </role>
  </access-config>
</access-security>
```

■ タグ説明

<accessor>	ロール情報を取得、更新するモジュールを設定します。 (複数指定可能)
<accessor-class>	ロール情報を取得するクラスを設定します。 RoleAccessor インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>	取得したロール情報を修飾するモジュールを設定します。 (複数指定可能)
<reader-decorator-class>	取得したロール情報を修飾するクラスを設定します。 RoleReaderDecorator インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>	ロール情報を更新する前に呼ばれるモジュールを設定します。 (複数指定可能)
<writer-listener-class>	ロール情報を更新する前に呼ばれるクラスを設定します。 RoleQueryWriterListener インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>	ロール情報を更新した後に呼ばれるモジュールを設定します。 (複数指定可能)
<writer-listener-class>	ロール情報を更新した後に呼ばれるクラスを設定します。 RolePostWriterListener インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.6.5 ロール情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.role になります。

- **RoleAccessor インタフェース**
  ロール情報を取得、更新するモジュールのインタフェース。

- **RoleReaderDecorator インタフェース**
  取得したロール情報を修飾するモジュールのインタフェース。

- **RoleQueryWriterListener インタフェース**
  ロール情報を更新する前に呼ばれるモジュールのインタフェース。

- **RolePostWriterListener インタフェース**
  ロール情報を更新した後に呼ばれるモジュールのインタフェース。

3.6.6 ロール情報操作実装クラス

- **jp.co.intra_mart.foundation.security.StandardRoleAccessor クラス**
  ログイングループの標準データソースに対してロール情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。

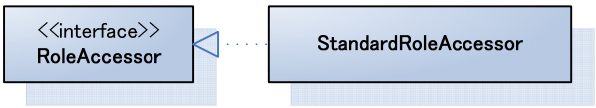


図 25 StandardRoleAccessor クラス

3.7 メニュー

3.7.1 メニュー情報

メニュー情報はログイングループに属します。(複数設定可能)

メニュー情報のメニューID はログイングループ内でユニークな値にする必要があります。

メニューは階層構造となります。

メニューには、フォルダとメニュー項目の2種類があります。

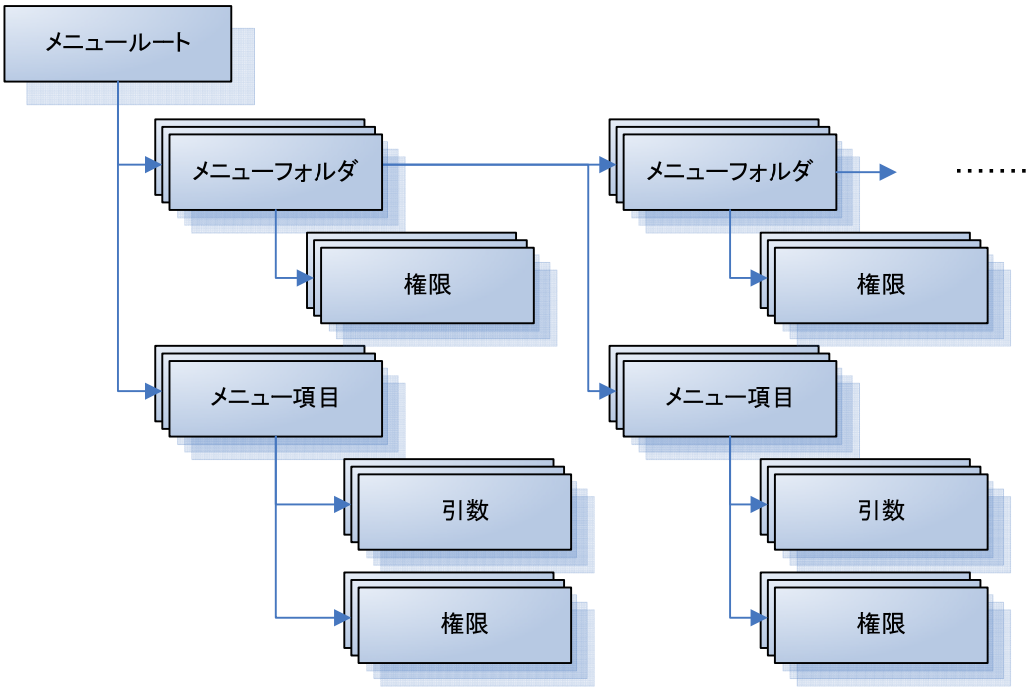


図 26 メニュー構成図

表 26 メニュー情報モデル

項目	説明	メニューフォルダ	メニュー項目	プロパティ(JavaScript)
メニューID	メニューID	○	○	menuId
メニュー名	メニューの表示名 [国際化]	○	○	displayName
ノードタイプ	メニューフォルダかメニュー項目かを指定	○	○	type
コンテンツの種類	内部 URL か外部 URL かを指定		○	contents
クライアントタイプ	クライアントタイプ		○	clientType
説明	メニューの説明			description
ソート番号	ソートするための番号			sortNumber
URL	表示するURL		○	url
引数	引き渡すパラメータ		○	argument

*○は必須項目です。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ メニュー親子関係

メニューの親子関係を表すモデルです。

表 27 メニュー親子関係情報モデル

項目	説明	プロパティ（JavaScript）
親メニューID	親メニューID	**parentMenuId**
子メニューID	子メニューID	**childMenuId**

■ メニュー引数

メニュー内容を表示する際に、リクエストに引き渡す引数です。

メニュー項目のみ有効です。

表 28 メニュー引数情報モデル

項目	説明
引数名	引数のキー名
引数値	引数の値

■ メニュー権限

メニューを表示するための権限です。（複数設定可能）

ログインユーザが保持する権限と同じ権限をもつメニュー表示するために用います。

メニューに設定できる権限は、以下の 4 種類です。

表 29 メニュー権限 ロールモデル

項目	説明
ロール ID	ロール ID

表 30 メニュー権限 組織モデル

項目	説明	プロパティ（JavaScript）
会社コード	会社コード	**companyCode**
組織コード	組織コード	**departmentCode**

表 31 メニュー権限 役職モデル

項目	説明	プロパティ（JavaScript）
会社コード	会社コード	**companyCode**
役職コード	役職コード	**postCode**

表 32 メニュー権限 パブリックグループモデル

項目	説明	プロパティ（JavaScript）
グループセットコード	グループセットコード	**groupSetCode**
グループコード	グループコード	**groupCode**

3.7.2 メニューのモデル

- **jp.co.intra_mart.foundation.security.menu.MenuItem クラス**
  メニュー情報を提供するクラスです。
  システムメニュー、管理メニュー、メニューで共通に利用します。
  メニュー情報は必ずこのクラスを通して操作します。

- **jp.co.intra_mart.foundation.security.menu.MenuInclusion クラス**
  メニュー情報の親子関係を提供するクラスです。
  システムメニュー、管理メニュー、メニューで共通に利用します。
  メニュー情報の親子関係は必ずこのクラスを通して操作します。

- **jp.co.intra_mart.foundation.security.DepartmentInfo クラス**
  メニュー権限の組織情報を提供するクラスです。
  メニュー権限の組織情報は必ずこのクラスを通して操作します。

- **jp.co.intra_mart.foundation.security.PostInfo クラス**
  メニュー権限の役職情報を提供するクラスです。
  メニュー権限の役職情報は必ずこのクラスを通して操作します。

- **jp.co.intra_mart.foundation.security.PublicGroupInfo クラス**
  メニュー権限のパブリックグループ情報を提供するクラスです。
  メニュー権限のパブリックグループ情報は必ずこのクラスを通して操作します。

3.7.3 メニュー情報へのアクセス

メニュー情報の取得は以下のように行います。

- スクリプト開発モデル
  MenuManager.getMenu メソッドを使用して取得します。

```
var manager = new MenuManager("default");
var menu = manager.getMenu("menu1");
```

上記の例では、ログイングループ ID（default）,メニューID（menu1）のメニュー情報を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.menu.MenuManager を用いて、メニュー情報を取得します。

```
MenuManager manager = new MenuManager("default");
MenuItem menu = manager.getMenu("menu1");
```

上記の例では、ログイングループ ID（default）,メニューID（menu1）のメニュー情報を取得します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.7.4 メニュー情報操作モジュール

実際にメニュー情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <menu>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>StandardMenuAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </menu>
  </access-config>
</access-security>
```

■ タグ説明

<accessor>	メニュー情報を取得、更新するモジュールを設定します。 （複数指定可能）
<accessor-class>	メニュー情報を取得するクラスを設定します。 MenuAccessor インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>	取得したメニュー情報を修飾するモジュールを設定します。 （複数指定可能）
<reader-decorator-class>	取得したメニュー情報を修飾するクラスを設定します。 MenuReaderDecorator インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>	メニュー情報を更新する前に呼ばれるモジュールを設定します。 （複数指定可能）
<writer-listener-class>	メニュー情報を更新する前に呼ばれるクラスを設定します。 MenuQueryWriterListener インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>	メニュー情報を更新した後に呼ばれるモジュールを設定します。 （複数指定可能）
<writer-listener-class>	メニュー情報を更新した後に呼ばれるクラスを設定します。 MenuPostWriterListener インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

3.7.5 メニュー情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.menu になります。

- **MenuAccessor インタフェース**
  メニュー情報を取得、更新するモジュールのインタフェース。

- **MenuReaderDecorator インタフェース**
  取得したメニュー情報を修飾するモジュールのインタフェース。

- **MenuQueryWriterListener インタフェース**
  メニュー情報を更新する前に呼ばれるモジュールのインタフェース。

- **MenuPostWriterListener インタフェース**
  メニュー情報を更新した後に呼ばれるモジュールのインタフェース。

3.7.6 メニュー情報操作実装クラス

- **jp.co.intra_mart.foundation.security.StandardMenuAccessor クラス**
  ログイングループの標準データソースに対してメニュー情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。

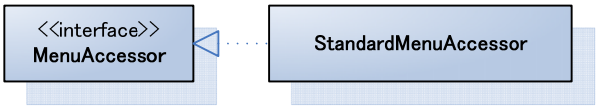


図 27 StandardMenuAccessor クラス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.8 バッチ

3.8.1 バッチ情報

バッチ情報はログイングループに属します。(複数設定可能)

バッチ情報のバッチ ID はログイングループ内でユニークな値にする必要があります。

表 33 バッチ情報モデル

項目	説明	プロパティ (JavaScript)
バッチ ID [必須]	バッチID	id
バッチ名 [必須]	バッチの表示名	name
実行プログラム言語 [必須]	実行プログラムのタイプ。 'JavaScript' または 'JAVA' です。	programLaguage
実行プログラムパス [必須]	実行プログラムパス。 言語が JavaScript である場合は、pages 以下のパスです。 言語が java である場合は、クラスのフルパッケージ名です。	programPath
実行年	実行する年 -1 を設定した場合は、毎年実行されます。	runYear
実行月	実行する月 -1 を設定した場合は、毎月実行されます。	runMonth
実行日	実行する日 -1 を設定した場合は、毎日実行されます。	runDay
実行曜日	実行する曜日 設定できる値は、0(日曜日)から 6(土曜日)までです。 上記の値以外を設定した場合は、-1 に設定されます。 この設定を無効にするには-1 を設定します。 実行日(runDay)が設定されいる場合は、この設定は無視されます。	runDayWeek
実行時	実行する時 -1 を設定した場合は、毎時実行されます。	runHour
実行分	実行する分 -1 を設定した場合は、毎分実行されます。	runMinute
実行秒	実行する秒 -1 を設定した場合は、毎秒実行されます。	runSecond
有効・無効フラグ	バッチが有効かどうか。	validate

3.8.2 バッチのモデル

- **jp.co.intra_mart.foundation.security.batchbatch クラス**

  バッチ情報を提供するクラスです。

  バッチ情報は必ずこのクラスを通して操作します。

3.8.3 バッチ情報へのアクセス

バッチ情報の取得は以下のように行います。

- スクリプト開発モデル

  BatchManager.getBatch メソッドを使用して取得します。

```
var manager = new BatchManager("default");
var batch = manager.getBatch("Batch1");
```

上記の例では、ログイングループ ID（default）,バッチ ID（Batch1）のバッチ情報を取得します。

- JavaEE 開発モデル

  jp.co.intra_mart.foundation.security.batch.BatchManager を用いて、バッチ情報を取得します。

```
BatchManager manager = new BatchManager("default");
Batch batch = manager.getBatch("Batch1");
```

上記の例では、ログイングループ ID（default）,バッチ ID（Batch1）のバッチ情報を取得します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.8.4 バッチ情報操作モジュール

実際にバッチ情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <batch>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>StandardBatchAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </batch>
  </access-config>
</access-security>
```

■ タグ説明

<accessor>	バッチ情報を取得、更新するモジュールを設定します。(複数指定可能)	
	<accessor-class>	バッチ情報を取得するクラスを設定します。 BatchAccessor インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>	取得したバッチ情報を修飾するモジュールを設定します。 (複数指定可能)	
	<reader-decorator-class>	取得したバッチ情報を修飾するクラスを設定します。 BatchReaderDecorator インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>	バッチ情報を更新する前に呼ばれるモジュールを設定します。 (複数指定可能)	
	<writer-listener-class>	バッチ情報を更新する前に呼ばれるクラスを設定します。 BatchQueryWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>	バッチ情報を更新した後に呼ばれるモジュールを設定します。 (複数指定可能)	
	<writer-listener-class>	バッチ情報を更新した後に呼ばれるクラスを設定します。 BatchPostWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

3.8.5 バッチ情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.Batch になります。

- **BatchAccessor インタフェース**
  バッチ情報を取得、更新するモジュールのインタフェース。

- **BatchReaderDecorator インタフェース**
  取得したバッチ情報を修飾するモジュールのインタフェース。

- **BatchQueryWriterListener インタフェース**
  バッチ情報を更新する前に呼ばれるモジュールのインタフェース。

- **BatchPostWriterListener インタフェース**
  バッチ情報を更新した後に呼ばれるモジュールのインタフェース。

3.8.6 バッチ情報操作実装クラス

- **jp.co.intra_mart.foundation.security.StandardBatchAccessor クラス**
  ログイングループの標準データソースに対してバッチ情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。

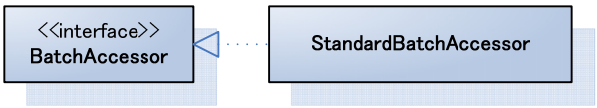


図 28 StandardBatchAccessor クラス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.9 アクセスコントローラ

ページ内のアクセスコントローラ制御内の項目をアクセス権により、表示、非表示行います。
アクセスコントローラ用のタグで囲まれた領域がアクセスコントローラの制御範囲となります。
アクセス権はロール、組織、パブリックグループ、役職により制御できます。
アクセス権が存在しないユーザに対しては、アクセスコントローラ制御内の項目を非表示にします。

3.9.1 アクセスコントローラ情報

アクセスコントローラ情報はログイングループに属します。
アクセスコントローラ情報のアクセスコントローラ ID はログイングループ内でユニークな値にする必要があります。

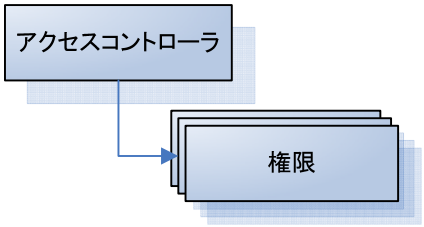


図 29 アクセスコントローラ構成図

表 34 アクセスコントローラ情報モデル

項目	説明	プロパティ (JavaScript)
アクセスコントローラ ID	アクセスコントローラID	controllerId
アクセスコントローラ名	アクセスコントローラの表示名	displayName
説明	アクセスコントローラの説明	description

■ アクセスコントローラ権限
アクセスコントローラの制御範囲を表示するための権限です。(複数設定可能)
ログインユーザが保持する権限と同じ権限をもつアクセスコントローラの制御範囲を表示するために用います。アクセスコントローラに設定できる権限は、以下の 4 種類です。

表 35 アクセスコントローラ権限 ロールモデル

項目	説明
ロール ID	ロール ID

表 36 アクセスコントローラ権限 組織モデル

項目	説明	プロパティ (JavaScript)
会社コード	会社コード	**companyCode**
組織コード	組織コード	**departmentCode**

表 37 アクセスコントローラ権限 役職モデル

項目	説明	プロパティ (JavaScript)
会社コード	会社コード	**companyCode**
役職コード	役職コード	**postCode**

表 38 アクセスコントローラ権限 パブリックグループモデル

項目	説明	プロパティ (JavaScript)
グループセットコード	グループセットコード	**groupSetCode**
グループコード	グループコード	**groupCode**

3.9.2 アクセスコントローラのモデル

- **jp.co.intra_mart.foundation.security.access.AccessController クラス**
  アクセスコントローラ情報を提供するクラスです。
  アクセスコントローラ情報は必ずこのクラスを通して操作します。

- **jp.co.intra_mart.foundation.security.DepartmentInfo クラス**
  アクセスコントローラ権限の組織情報を提供するクラスです。
  アクセスコントローラ権限の組織情報は必ずこのクラスを通して操作します。

- **jp.co.intra_mart.foundation.security.PostInfo クラス**
  アクセスコントローラ権限の役職情報を提供するクラスです。
  アクセスコントローラ権限の役職情報は必ずこのクラスを通して操作します。

- **jp.co.intra_mart.foundation.security.PublicGroupInfo クラス**
  アクセスコントローラ権限のパブリックグループ情報を提供するクラスです。
  アクセスコントローラ権限のパブリックグループ情報は必ずこのクラスを通して操作します。

3.9.3 アクセスコントローラ情報へのアクセス

アクセスコントローラ情報の取得は以下のように行います。

- スクリプト開発モデル
  AccessControllerManager.getAccessContoroler メソッドを使用して取得します。

```
var manager = new AccessControllerManager("default");
var controller = manager.getAccessController("controller1");
```

上記の例では、ログイングループ ID(default),アクセスコントローラ ID(controller1)のアクセスコントローラ情報を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.access.AccessControllerManager を用いて、アクセスコントローラ情報を取得します。

```
AccessControllerManager manager = new AccessControllerManager ("default");
AccessController controller = manager.getAccessController("controller1");
```

上記の例では、ログイングループ ID(default),アクセスコントローラ ID(controller1)のアクセスコントローラ情報を取得します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.9.4 アクセスコントローラ制御タグ

アクセスコントローラの機能を利用するには、カスタムタグを利用します。
このタグで囲まれた領域がアクセスコントローラの制御範囲となります。

■ スクリプト開発モデル用カスタムタグ

タグ名	accessCtrl	
属性	controller	アクセスコントローラ ID を指定する。
属性	defaultShow	アクセスコントローラ ID の情報定義が存在しない場合の表示状態。 true : 表示 false : 非表示 省略した場合のデフォルトは”false”

```
<IMART type=”accessCtrl” controller=”アクセスコントローラ ID”>

    表示/非表示の制御を行いたい内容

</IMART>
```

■ JavaEE 開発モデル用カスタムタグ

タグ名	accessCtrl	
属性	controller	アクセスコントローラ ID を指定する。
属性	defaultShow	アクセスコントローラ ID の情報定義が存在しない場合の表示状態。 true : 表示 false : 非表示 省略した場合のデフォルトは”false”

```
<imart:accessCtrl controller=”アクセスコントローラ ID”>

    表示/非表示の制御を行いたい内容

</imart:accessCtrl>
```

3.9.5 アクセスコントローラ情報操作モジュール

実際にアクセスコントローラ情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

```
<access-security>
  <access-config>
    <access-controller>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>StandardAccessControllerAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
    </access-controller>
  </access-config>
</access-security>
```

■ タグ説明

<accessor>		アクセスコントローラ情報を取得、更新するモジュールを設定します。 （複数指定可能）
	<accessor-class>	アクセスコントローラ情報を取得するクラスを設定します。 AccessControllerAccessor インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>		取得したアクセスコントローラ情報を修飾するモジュールを設定します。 （複数指定可能）
	<reader-decorator-class>	取得したアクセスコントローラ情報を修飾するクラスを設定します。 AccessControllerReaderDecorator インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>		アクセスコントローラ情報を更新する前に呼ばれるモジュールを設定します。 （複数指定可能）
	<writer-listener-class>	アクセスコントローラ情報を更新する前に呼ばれるクラスを設定します。 AccessControllerQueryWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>		アクセスコントローラ情報を更新した後に呼ばれるモジュールを設定します。 （複数指定可能）
	<writer-listener-class>	アクセスコントローラ情報を更新した後に呼ばれるクラスを設定します。 AccessControllerPostWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.9.6 アクセスコントローラ情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.access になります。

- **AccessControllerAccessor インタフェース**
  アクセスコントローラ情報を取得、更新するモジュールのインタフェース。

- **AccessControllerReaderDecorator インタフェース**
  取得したアクセスコントローラ情報を修飾するモジュールのインタフェース。

- **AccessControllerQueryWriterListener インタフェース**
  アクセスコントローラ情報を更新する前に呼ばれるモジュールのインタフェース。

- **AccessControllerPostWriterListener インタフェース**
  アクセスコントローラ情報を更新した後に呼ばれるモジュールのインタフェース。

3.9.7 アクセスコントローラ情報操作実装クラス

- **jp.co.intra_mart.foundation.security.StandardAccessControllerAccessor クラス**
  ログイングループの標準データソースに対してアクセスコントローラ情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。

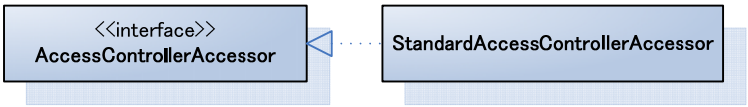


図 30 StandardAccessControllerAccessor クラス

3.10 Webサービスアクセス

Web サービスクライアントから要求されたオペレーションのアクセス権限を制御します。
アクセス権はロールにより制御できます。
アクセス権が存在しないユーザに対しては、オペレーションの実行ができません。

3.10.1 Webサービスアクセス情報

Web サービスアクセス情報はログイングループに属します。
Web サービスアクセス情報のWebサービス名とオペレーションはログイングループ内でユニークな値にする必要があります。

表 39 アクセスコントローラ情報モデル

項目	説明	プロパティ（JavaScript）
Webサービス名	Webサービス名	wsName
オペレーション	オペレーション	wsOperation
説明	Web サービスアクセスの説明	description

■ Web サービスアクセス権限
アクセスコントローラの制御範囲を表示するための権限です。（複数設定可能）
ログインユーザが保持する権限と同じ権限をもつアクセスコントローラの制御範囲を表示するために用います。
Web サービスアクセスに設定できる権限は、以下の権限です。

表 40 Web サービスアクセス権限 ロールモデル

項目	説明
ロール ID	ロール ID

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.10.2 Webサービスアクセスのモデル

- **jp.co.intra_mart.foundation.security.web_service.WSAccess クラス**
  Web サービスアクセス情報を提供するクラスです。
  Web サービスアクセス情報は必ずこのクラスを通して操作します。

3.10.3 Webサービスアクセス情報へのアクセス

Web サービスアクセス情報の取得は以下のように行います。

- スクリプト開発モデル
  WSAccessManagerObject.getAccessContoroler メソッドを使用して取得します。

```
var manager = new WSAccessManager("default");
var controller = manager.getWSAccess("service1", "operation1");
```

上記の例では、ログイングループ ID（default）,Web サービス名（service1）,オペレーション(operation1)の Web サービスアクセス情報を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.web_service.WSAccessManager を用いて、アクセスコントローラ情報を取得します。

```
WSAccessManager manager = new WSAccessManager ("default");
AccessController controller = manager.getWSAccess("service1", "operation1");
```

上記の例では、ログイングループ ID（default）,Web サービス名（service1）,オペレーション(operation1)の Web サービスアクセス情報を取得します。

3.10.4 Webサービスアクセス情報操作モジュール

実際にWebサービスアクセス情報へアクセスするモジュールは、設定ファイルに記述します。

設定ファイルは、サーバマネージャ/conf/ws-access-config.xml です。

```
<ws-access-config>
  <access-config>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>StandardWSAccessAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
  </access-config>

  <backup-config>
    <backup-group>
      <category>standard</category>
      <import>
        <import-class>SampleInpoter</import-class>
      </import>
      <export>
        <export-class>SampleExporter</export-class>
      </export>
    </backup-group>
  </backup-config>
</ws-access-config>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ タグ説明

<access-config>			Webサービスアクセス情報操作機能の設定です。
	<accessor>		Webサービスアクセス情報を取得するモジュールの設定します。
		<acsessor-class>	Webサービスアクセス情報を、更新するモジュールの実装クラスを設定します。（複数指定可能） jp.co.intra_mart.foundation.security.web_service.WSAccessAccessor を実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。
	<reader-decorator>		取得したWebサービスアクセス情報を修飾するモジュールを設定します。（複数指定可能）
		<reader-decorator-class>	取得したWebサービスアクセス情報を修飾するクラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserReaderDecorator インタフェースを実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。
	<query-writer-listener>		Webサービスアクセス情報を更新する前に呼ばれるモジュールを設定します。（複数指定可能）
		<writer-listener-class>	Webサービスアクセス情報を更新する前に呼ばれるクラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserQueryWriterListener インタフェースを実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。
	<post-writer-listener>		Webサービスアクセス情報を更新した後に呼ばれるモジュールを設定します。（複数指定可能）
		<writer-listener-class>	Webサービスアクセス情報を更新した後に呼ばれるクラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserPostWriterListener インタフェースを実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<backup-config>				Webサービスアクセス情報のインポート/エクスポート機能の設定です。
	<backup-group>			インポート/エクスポートのグループ設定です。（複数設定可能）
		<category>		インポート/エクスポートのグループの名称です。 **<backup-config>内でユニークな値です。**
		<import>		インポートに関する設定です。
			<import-class>	Webサービスアクセス情報のインポート実装クラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserImporter を実装する必要があります。
			<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。
		<export>		エクスポートに関する設定です。
			<export-class>	Webサービスアクセス情報のエクスポート実装クラスを設定します。 jp.co.intra_mart.foundation.security.adminuser.AdminUserExporterAccessor を実装する必要があります。
			<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

3.10.5 Webサービスアクセス情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.web_service になります。

- **WSAccessAccessor インタフェース**
  Webサービスアクセス情報を取得、更新するモジュールのインタフェース。

- **WSAccessReaderDecorator インタフェース**
  取得したWebサービスアクセス情報を修飾するモジュールのインタフェース。

- **WSAccessQueryWriterListener インタフェース**
  Webサービスアクセス情報を更新する前に呼ばれるモジュールのインタフェース。

- **WSAccessPostWriterListener インタフェース**
  Webサービスアクセス情報を更新した後に呼ばれるモジュールのインタフェース。

3.10.6 Webサービスアクセス情報操作実装クラス

- **jp.co.intra_mart.foundation.security.adminuser.StandardWSAccessAccessor クラス**
  ログイングループの標準データソースに対して Web サービスアクセス情報を取得、更新するモジュールの実装クラス。
  intra-mart WebPlatform 標準の実装クラスです。

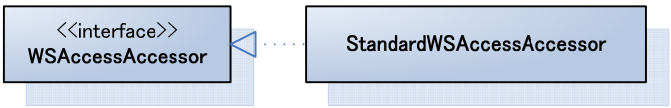


図 31 **StandardWSAccessAccessor クラス**

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.11 カレンダー

3.11.1 カレンダー表示情報

カレンダーはログイングループに属します。(複数設定可能)
カレンダーのカレンダーID はログイングループ内でユニークな値にする必要があります。
カレンダーはカレンダー表示情報を複数持つことが可能です。
カレンダー表示情報のデータ ID は1つのカレンダー内でユニークな値にする必要があります。
カレンダーID がログイングループ ID を同じものを、ログイングループの標準のカレンダーとして扱います。

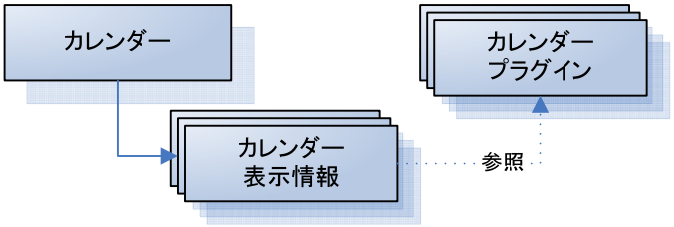


図 32 カレンダー構成図

表 41 カレンダー表示情報モデル

項目	説明	プロパティ (JavaScript)
データ ID [必須]	カレンダーデータID	id
データ タイプ [必須]	カレンダーデータのタイプ 設定できる値は、 'config'(名称)、'holiday'(休日)、 'weekday'(非休日)、'nationalHoliday'(公休日)のいずれかです。 上記以外の値が設定された場合は、 'config'(名称)に設定されます。	type
データ名 [必須]	カレンダーデータの名称	name
表示名 [必須]	表示名 [国際化]	displayName
説明	カレンダー情報の説明	description
年	年 この設定を無効にするには-1 を設定します。	year
月	月 この設定を無効にするには-1 を設定します。	month
日	日 この設定を無効にするには-1 を設定します。	day
週	第 n 週目 この設定を無効にするには-1 を設定します。	week
次	第 n～曜日 この設定を無効にするには-1 を設定します。	times
曜日	曜日 設定した曜日(日)を特定したい場合は、第n～曜日の設定(times)または第n週の設定(week)と共に設定してください。 第n～曜日の設定、第n週の設定、共に設定を行わなかった場合は、すべての指定曜日の設定となります。 この設定を無効にするには-1 を設定します。	dayWeek
有効開始年	有効開始年	validStartDate
有効終了年	有効終了年	validEndDate
プラグイン関数	拡張関数設定	plugin
色	表示色	color

3.11.2 カレンダー表示情報のモデル

- **jp.co.intra_mart.foundation.security.calendarCalendarInfo クラス**
  カレンダー情報を提供するクラスです。
  カレンダー情報は必ずこのクラスを通して操作します。

3.11.3 カレンダー表示情報へのアクセス

カレンダー表示情報の取得は以下のように行います。

- スクリプト開発モデル
  CalendarManager.getCalendarInfo メソッドを使用して取得します。

```
var manager = new CalendarManager("default");
var cInfo = manager.getCalendarInfo("default""cInfo1");
```

上記の例では、ログイングループ ID（default），カレンダーID（calendar1）,カレンダー表示情報 ID（cInfo1）のカレンダー情報を取得します。

- JavaEE 開発モデル
  jp.co.intra_mart.foundation.security.calendar.CalendarManager を用いて、カレンダー情報を取得します。

```
CalendarManager manager = new CalendarManager("default");
CalendarInfo cInfo = manager. getCalendarInfo("default""cInfo1");
```

上記の例では、ログイングループ ID（default），カレンダーID（calendar1）,カレンダー表示情報 ID（cInfo1）のカレンダー情報を取得します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.11.4 カレンダー表示情報操作モジュール

実際にカレンダー情報へアクセスするモジュールは、設定ファイルに記述します。

```
設定ファイルは、サーバマネージャ/conf/access-security.xml です。
<access-security>
  <access-config>
    <calendar>
      <reader-decorator>
        <reader-decorator-class>SampleReaderDecorator</reader-decorator-class>
      </reader-decorator>
      <accessor>
        <accessor-class>StandardCalendarAccessor</accessor-class>
        <init-param>
          <param-name>ParamName</param-name>
          <param-value>ParamValue</param-value>
        </init-param>
      </accessor>
      <query-writer-listener>
        <writer-listener-class>SampleQueryWriterListener</writer-listener-class>
      </query-writer-listener>
      <post-writer-listener>
        <writer-listener-class>SamplePostWriterListener</writer-listener-class>
      </post-writer-listener>
      <plugin>
        <plugin-id>plugin-id</plugin-id>
        <plugin-class>plugin-class</plugin-class>
      </plugin>
    </calendar>
  </access-config>
</access-security>
```

■ タグ説明

<accessor>	カレンダー表示情報を取得、更新するモジュールを設定します。 （複数指定可能）
<accessor-class>	カレンダー表示情報を取得するクラスを設定します。 CalendarControllerAccessor インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<reader-decorator>	取得したカレンダー表示情報を修飾するモジュールを設定します。 （複数指定可能）
<reader-decorator-class>	取得したカレンダー表示情報を修飾するクラスを設定します。 CalendarControllerReaderDecorator インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<query-writer-listener>	カレンダー表示情報を更新する前に呼ばれるモジュールを設定します。 （複数指定可能）
<writer-listener-class>	カレンダー表示情報を更新する前に呼ばれるクラスを設定します。 CalendarControllerQueryWriterListener インタフェースを実装する必要があります。
<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<post-writer-listener>		カレンダー表示情報を更新した後に呼ばれるモジュールを設定します。 （複数指定可能）
	<writer-listener-class>	カレンダー表示情報を更新した後に呼ばれるクラスを設定します。 CalendarControllerPostWriterListener インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。

<plugin>		カレンダー表示情報の有効無効を決定する拡張プラグインクラスをを設定します。 （複数指定可能）
	<plugin-id>	拡張プラグインの ID を設定します。 calendar タグ内でユニークな ID を指定します。
	<plugin-class>	拡張プラグインの実装クラスを設定します。 CalendarPlugin インタフェースを実装する必要があります。

3.11.5 カレンダー表示情報操作インタフェース

パッケージは jp.co.intra_mart.foundation.security.calendar になります。

- **CalendarAccessor インタフェース**
  カレンダー表示情報を取得、更新するモジュールのインタフェース。

- **CalendarReaderDecorator インタフェース**
  取得したカレンダー表示情報を修飾するモジュールのインタフェース。

- **CalendarQueryWriterListener インタフェース**
  カレンダー表示情報を更新する前に呼ばれるモジュールのインタフェース。

- **CalendarPostWriterListener インタフェース**
  カレンダー表示情報を更新した後に呼ばれるモジュールのインタフェース。

- **CalendarPlugin インタフェース**
  カレンダー表示情報拡張プラグイン実装クラスに実装するインタフェース。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.11.6 カレンダー表示情報操作実装クラス

■ **jp.co.intra_mart.foundation.security.StandardCalendarAccessor** クラス

ログイングループの標準データソースに対してカレンダー表示情報を取得、更新するモジュールの実装クラス。

intra-mart WebPlatform 標準の実装クラスです。

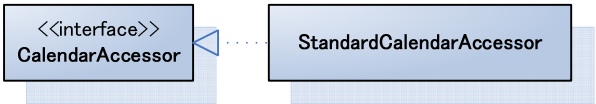


図 33 StandardCalendarAccessor クラス

■ **jp.co.intra_mart.foundation.security.calendar.CalendarPluginAutumnalEquinox** クラス

秋分の日を判定する拡張プラグイン実装クラス。

intra-mart WebPlatform 標準の実装クラスです。

プラグイン ID は **AutumnalEquinox**

図 34 CalendarPluginAutumnalEquinox クラス

■ **jp.co.intra_mart.foundation.security.calendar.CalendarPluginEndOfTheMonth** クラス

月末を判定する拡張プラグイン実装クラス。

intra-mart WebPlatform 標準の実装クラスです。

プラグイン ID は **EndOfTheMonth**

図 35 CalendarPluginEndOfTheMonth クラス

■ **jp.co.intra_mart.foundation.security.calendar.CalendarPluginLastDayInTheMonth** クラス

月内の最終曜日を判定する拡張プラグイン実装クラス。

intra-mart WebPlatform 標準の実装クラスです。

プラグイン ID は **LastDayInTheMonth**

図 36 CalendarPluginLastDayInTheMonth クラス

■ **jp.co.intra_mart.foundation.security.calendar.CalendarPluginMondayMakeupHoliday** クラス

振り替え休日を判定する拡張プラグイン実装クラス。

intra-mart WebPlatform 標準の実装クラスです。

プラグイン ID は **MondayMakeupHoliday**

図 37 CalendarPluginMondayMakeupHoliday クラス

■ **jp.co.intra_mart.foundation.security.calendar.CalendarPluginVernalEquinox** クラス

春分の日を判定する拡張プラグイン実装クラス。

intra-mart WebPlatform 標準の実装クラスです。

プラグイン ID は **VernalEquinox**

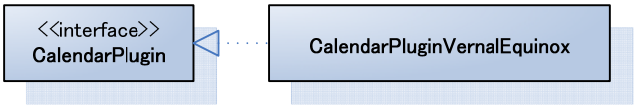


図 38 CalendarPluginVernalEquinox クラス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.12 情報更新管理

情報更新管理では、マスタデータなどの情報の最終更新日を管理できます。

マスタデータをメモリ上でプールする場合、マスタデータの更新と同時に modify メソッドを利用して、更新管理を行っておくことにより、getLastModified メソッドを用いて効果的にメモリ上にプールされているデータを更新できます。
データの種類によってユニークな更新管理キーを用いて管理できます。

なお、更新管理キーが 'im_' から始まるキーは、システム用として予約されています。
上記のキーを用いての更新系 API（modfiy,remove など)を使用しないでください。

ログイングループ単位での管理が対象となります。

3.12.1 情報更新管理マネージャ

情報更新管理マネージャは、2 つの開発モデルで提供されています。

スクリプト開発モデル	**UpdateManager**
JavaEE 開発モデル	**jp.co.intra_mart.foundation.update.UpdateManager**

機能は、以下の通りです。

- 更新管理キーに対する更新日付の保管。
- 更新管理キーによる最終更新日付の取得。
- 更新が行われたかのどうかのチェック。

3.12.2 既存の更新管理情報

以下のアクセスセキュリティの情報に対しては、更新管理が実装されています。
()内は、更新管理キー。

- アカウント（im_account）
- メニュー（im_menu）
- カレンダー（im_calendar_*）
- バッチ（im_batch）
- アクセスコントローラ（im_controller）
- ロール（im_role）

上記の情報が更新される時と同時に、更新管理マネージャを使って、最終更新日付を保管しています。
また、各マネージャの isUpDate 関数で、情報更新管理マネージャの isUpdate 関数を利用しています。

3.13 インポート／エクスポート

アクセスセキュリティ情報をインポート／エクスポートする機能を提供します。

カテゴリ別にさまざまなインポート／エクスポート機能を実装することが可能です。

intra-mart WebPlatform 標準では、カテゴリ(standard)で標準のインポート／エクスポート機能（XML 入出力）を実装しています。

実際にインポート／エクスポートを行う実装クラスは、設定ファイルにて行います。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

インポート/エクスポート可能な情報は以下の通りです。

- 管理メニュー
- アカウント
- ロール
- メニュー
- バッチ
- カレンダー

インポートファイルのフォーマットについては ServerManager/specification/dtd を参照してください。

3.13.1 インポート／エクスポートのメソッド

インポート／エクスポートのメソッドは情報別に各 Manager クラスに実装されています。

表 42 インポート／エクスポートのメソッド一覧

情報	インポートメソッド	エクスポートメソッド
管理メニュー	AdminMenuManager.importData	AdminMenuManager.exportData
アカウント	AccountManager.importData	AccountManager.exportData
ロール	RoleManager.importData	RoleManager.exportData
メニュー	MenuManager.importData	MenuManager.exportData
バッチ	BatchManager.importData	BatchManager.exportData
カレンダー	CalendarManager.importData	CalendarManager.exportData

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.13.2 インポート／エクスポートの設定

インポート／エクスポートの設定は設定ファイルで行います。

設定ファイルは、サーバマネージャ/conf/access-security.xml です。

backup-config での例を以下に示します。

```
<access-security>
  <backup-config>
    <backup-group>
      <category>Category</category>
      <admin-menu-backup>
        <import>
          <import-class>AdminMenuImportClass</import-class>
            <init-param>
             <param-name>ParamName</param-name>
             <param-value>ParamValue</param-value>
            </init-param>
        </import>
        <export>
          <export-class>AdminMenuExportClass</export-class>
        </export>
      </admin-menu-backup>

      <account-backup>
        ・・・・
      </account-backup>

      <controller-backup>
        ・・・・
      </controller-backup>

      <role-backup>
        ・・・・
      </role-backup>

      <menu-backup>
        ・・・・
      </menu-backup>

      <calendar-backup>
        ・・・・
      </calendar-backup>

      <batch-backup>
        ・・・・
      </batch-backup>

    </backup-group>
  </backup-config>
</access-security>
```

■ タグの説明

<backup-group>

インポート／エクスポートをグループ化するタグです。（複数指定可能）
のタグ内、カテゴリ名と各情報別にインポート／エクスポート実装クラスを設定します。

<category>

このパーミッショングループにカテゴリ名設定します。

各情報別タグ

各情報別にタグが用意されています。

情報	タグ名
管理メニュー	admin-menu-backup
アカウント	account-backup
アクセスコントローラ	controller-backup
ロール	role-backup
メニュー	menu-backup
バッチ	batch-backup
カレンダー	calendar-backup

<import>

インポート実装クラスの設定を行います。

<import-class>

インポート用の実装クラスをを設定します。

各情報別に実装するインタフェースが異なります。

情報	実装するインタフェース
管理メニュー	AdminMenuImporter
アカウント	AccountImporter
アクセスコントローラ	AccessControllerImporter
ロール	RoleImporter
メニュー	MenuImporter
バッチ	BatchImporter
カレンダー	CalendarImporter

<init-param>

このクラスに渡す初期化パラメータです。(複数指定可能)
このクラスが生成された時に一度だけこのパラメータで初期化します。

<export>

エクスポート実装クラスの設定を行います。

<export-class>

エクスポート用の実装クラスをを設定します。

各情報別に実装するインタフェースが異なります。

情報	実装するインタフェース
管理メニュー	AdminMenuExporter
アカウント	AccountExporter
アクセスコントローラ	AccessControllerExporter
ロール	RoleExporter
メニュー	MenuExporter
バッチ	BatchExporter
カレンダー	CalendarExporter

<init-param>

このクラスに渡す初期化パラメータです。(複数指定可能)
このクラスが生成された時に一度だけこのパラメータで初期化します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.13.3 インポート／エクスポートの呼出し

アカウントのインポートの呼出しは以下のように行います。

■ スクリプト開発モデル

AccountManager.importData メソッドを使用してインポートを行います。

```
var content = new VirtualFile("import.xml").load();
var manager = new AccountManager("default");
var result = manager.importData("standard", content);
```

上記の例では、ログイングループ ID（default）に対して,カテゴリ‘standard’を指定して、ストレージサービス直下の”import.xml”のデータをインポートします。

■ JavaEE 開発モデル

jp.co.intra_mart.foundation.securityAccount.AccountManager を用いて、インポートを行います。

```
InputStream in = new FileInputStream("C:/import.xml");

AccountManager manager = new AccountManager("default");
manager. importData("standard",in);
```

上記の例では、ログイングループ ID（default）に対して,カテゴリ‘standard’を指定して、”C:¥import.xml”のデータをインポートします。

3.13.4 インポート／エクスポートのインタフェース

カテゴリ	パッケージ	インポートインタフェース エクスポートインタフェース
管理メニュー	jp.co.intra_mart.foundation.security.men	AdminMenuImporter AdminMenuExporter
アカウント	jp.co.intra_mart.foundation.security.account	AccountImporter AccountExporter
アクセスコントローラ	jp.co.intra_mart.foundation.security.access	AccessControllerImporter AccessControllerExporter
ロール	jp.co.intra_mart.foundation.security.role	RoleImporter RoleExporter
メニュー	jp.co.intra_mart.foundation.security.menu	MenuImporter MenuExporter
カレンダー	jp.co.intra_mart.foundation.security.calendar	CalendarImporter CalendarExporter
バッチ	jp.co.intra_mart.foundation.security.batch	BatchImporter BatchExporter
グループ管理権限ユーザメニュー	jp.co.intra_mart.foundation.security.adminuser	AdminUserImporter AdminUserExporter

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

3.13.5 インポート／エクスポートの実装クラス

■ **jp.co.intra_mart.foundation.security.menu.StandardAdminMenuBackup** クラス

管理メニューのインポート／エクスポートの実装クラス。

入出力を XML ファイルで行います。

intra-mart WebPlatform 標準のクラスです。

カテゴリは **standard**

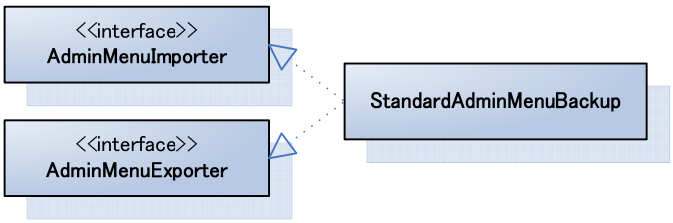


図 39 StandardAdminMenuBackup クラス

◆ StandardAdminMenuBackup クラスのパラメータ

StandardAdminMenuBackup クラスでは、エクスポートファイルのエンコーディングを変更できます。

表 43 StandardAdminMenuBackup クラスのパラメータの一覧

パラメータ名	説明
encoding	エクスポートファイルのエンコーディングを設定。 設定しない場合は、UTF-8 で出力します。

■ **jp.co.intra_mart.foundation.security.account.StandardAccountBackup** クラス

アカウントのインポート／エクスポートの実装クラス。

入出力を XML ファイルで行います。

intra-mart WebPlatform 標準のクラスです。

カテゴリは **standard**

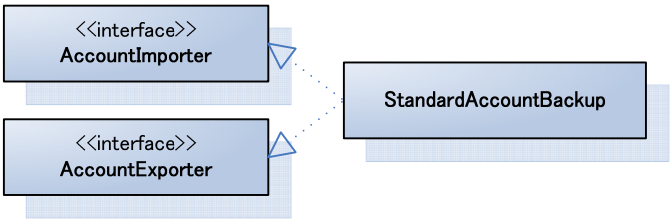


図 40 StandardAccountBackup クラス

◆ StandardAccountBackup クラスのパラメータ

StandardAccountBackup クラスでは、エクスポートファイルのエンコーディングを変更できます。

表 44 StandardAccountBackup クラスのパラメータの一覧

パラメータ名	説明
encoding	エクスポートファイルのエンコーディングを設定。 設定しない場合は、UTF-8 で出力します。
cryption-class	パスワード、およびモバイルパスワードを暗号化して保存するためのクラスを指定します。 設定するクラスは、jp.co.intra_mart.foundation.security.cryption.Cryption インタフェースを実装する必要があります。 省略した場合は、intra-mart WebPlatform 標準の暗号化アルゴリズムで暗号化します。 標準の暗号化アルゴリズムの実装クラスは jp.co.intra_mart.foundation.security.cryption.StandardCryption です。 また、暗号化を行わない実装クラスとして jp.co.intra_mart.foundation.security.cryption.NoCryption が用意されています。 暗号化複合化クラスを指定する場合は、access-security/backup-config/backup-group/account-backup/export にて以下のように記述して指定してください。 --- access-security.xml 内の設定例 --- <export> <export-class>jp.co.intra_mart.foundation.security.account.StandardAccountBackup </export-class> <init-param> <param-name>cryption-class</param-name> <param-value>設定する暗号化複合化クラス</param-value> </init-param> </export>

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ **jp.co.intra_mart.foundation.security.account.CsvAccountBackup** クラス

アカウントのインポート／エクスポートの実装クラス。

入出力を CSV ファイルで行います。

intra-mart WebPlatform 標準のクラスです。

カテゴリは **csv**

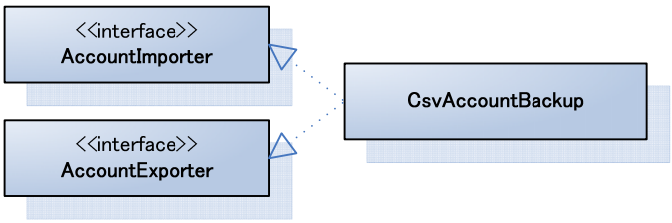


図 41 CsvAccountBackup クラス

■ **jp.co.intra_mart.foundation.security.access.StandardAccessControllerBackup** クラス

アクセスコントローラののインポート／エクスポートの実装クラス。

入出力を XML ファイルで行います。

intra-mart WebPlatform 標準のクラスです。

カテゴリは **standard**

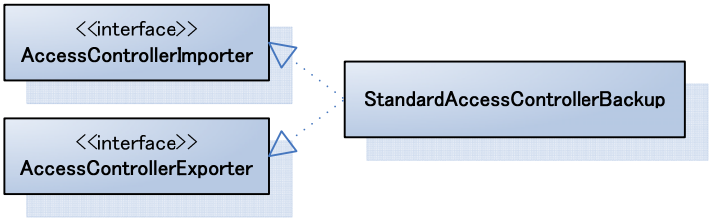


図 42 StandardAccessControllerBackup クラス

◆ StandardAccessControllerBackup クラスのパラメータ

StandardAccessControllerBackup クラスでは、エクスポートファイルのエンコーディングを変更できます。

表 45 StandardAccessControllerBackup クラスのパラメータの一覧

パラメータ名	説明
encoding	エクスポートファイルのエンコーディングを設定。 設定しない場合は、UTF-8 で出力します。

■ **jp.co.intra_mart.foundation.security.role.StandardRoleBackup** クラス

ロールのインポート／エクスポートの実装クラス。

入出力を XML ファイルで行います。

intra-mart WebPlatform 標準のクラスです。

カテゴリは **standard**

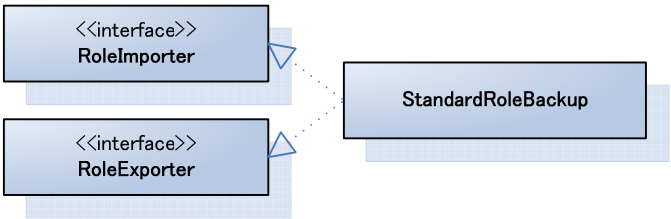


図 43 StandardRoleBackup クラス

◆ StandardRoleBackup クラスのパラメータ

StandardRoleBackup クラスでは、エクスポートファイルのエンコーディングを変更できます。

表 46 StandardRoleBackup クラスのパラメータの一覧

パラメータ名	説明
encoding	エクスポートファイルのエンコーディングを設定。 設定しない場合は、UTF-8 で出力します。

■ **jp.co.intra_mart.foundation.security.menu.StandardMenuBackup** クラス

メニューのインポート／エクスポートの実装クラス。

入出力を XML ファイルで行います。

intra-mart WebPlatform 標準のクラスです。

カテゴリは **standard**

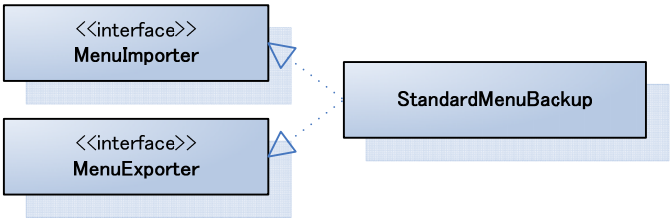


図 44 StandardMenuBackup クラス

◆ StandardMenuBackup クラスのパラメータ

StandardMenuBackup クラスでは、エクスポートファイルのエンコーディングを変更できます。

表 47 StandardMenuBackup クラスのパラメータの一覧

パラメータ名	説明
encoding	エクスポートファイルのエンコーディングを設定。 設定しない場合は、UTF-8 で出力します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ **jp.co.intra_mart.foundation.security.calendar.StandardCalendarBackup クラス**

カレンダーのインポート／エクスポートの実装クラス。

入出力を XML ファイルで行います。

intra-mart WebPlatform 標準のクラスです。

カテゴリは **standard**

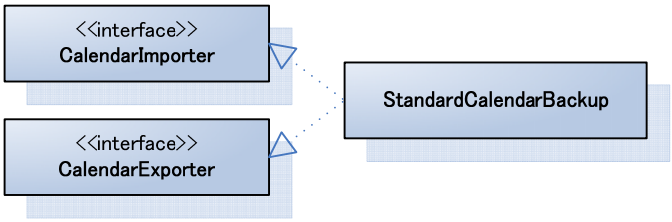


図 45 StandardCalendarBackup クラス

◆ StandardMenuBackup クラスのパラメータ

StandardMenuBackup クラスでは、エクスポートファイルのエンコーディングを変更できます。

表 48 StandardMenuBackup クラスのパラメータの一覧

パラメータ名	説明
encoding	エクスポートファイルのエンコーディングを設定。 設定しない場合は、UTF-8 で出力します。

■ **jp.co.intra_mart.foundation.security.batch.StandardBatchBackup クラス**

バッチのインポート／エクスポートの実装クラス。

入出力を XML ファイルで行います。

intra-mart WebPlatform 標準のクラスです。

カテゴリは **standard**

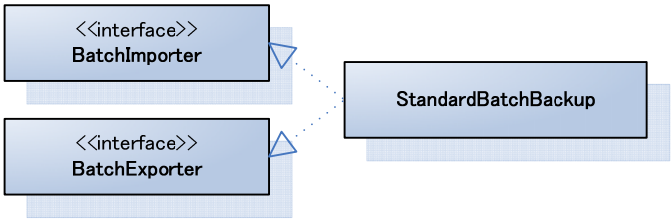


図 46 StandardBatchBackup クラス

◆ StandardBatchBackup クラスのパラメータ

StandardBatchBackup クラスでは、エクスポートファイルのエンコーディングを変更できます。

表 49 StandardBatchBackup クラスのパラメータの一覧

パラメータ名	説明
encoding	エクスポートファイルのエンコーディングを設定。 設定しない場合は、UTF-8 で出力します。

3.14 アクセスセキュリティの標準実装について

3.14.1 標準実装における情報の保管場所

アクセスセキュリティ情報の intra-mart WebPlatform 標準実装における保管場所は以下のようになっています。

表 50 アクセスセキュリティの標準実装

アクセスセキュリティ情報	保管場所	特記事項
対応ロケール	設定ファイル	サーバマネージャ/conf/system.xml サーバマネージャ/conf/i18n/locale_*.properties
対応クライアント	設定ファイル	サーバマネージャ/conf/system.xml サーバマネージャ/conf/i18n/client_*.properties
対応カラーパターン	設定ファイル	サーバマネージャ/conf/system.xml サーバマネージャ/conf/i18n/color_*.properties
システムデータベース	設定ファイル	サーバマネージャ/conf/data-source.xml
メッセージ	設定ファイル	サーバマネージャ/conf/message/*.properties
システムメニュー	設定ファイル	サーバマネージャ/conf/system-menu.xml
システム管理者	Permanent Data Service	
管理メニュー	Permanent Data Service	
ログイングループ	Permanent Data Service	
グループデータベース	設定ファイル	サーバマネージャ/conf/data-source.xml
グループ管理者	Permanent Data Service	
グループ管理権限 ユーザメニュー	ログイングループのデータベース	グループ管理権限のフラグは、アカウント属性に保管
アカウント	ログイングループのデータベース	アカウントのパスワードを暗号化
ロール	ログイングループのデータベース	
メニュー	ログイングループのデータベース	
カレンダー	ログイングループのデータベース	
バッチ	ログイングループのデータベース	
ライセンス	Permanent Data Service	ログイングループの MAX ライセンス情報 アカウントのライセンス情報 アカウントに対するアプリケーションロール情報

3.14.2 標準実装における注意点

アクセスセキュリティ情報の中で、ログイングループに属するグループ管理権限ユーザメニュー、アカウント、アクセスコントローラ、ロール、メニュー、カレンダー、バッチはデータベースに保管されます。

これらの情報について更新 API を用いて更新する場合、トランザクション内で更新処理を行ってください。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4 認証処理

4.1 ログインユーザ

ログイン可能なユーザは大きく分けてシステム管理者、グループ管理者、ユーザの 3 種類になります。
各ユーザでログインを行うための URL は以下のように異なります。

表 51 ログインユーザ別 URL

ログインユーザ	URL
システム管理者	http://サーバ名:ポート/imart/system.admin
グループ管理者	http://サーバ名:ポート/imart/ログイングループ ID.manager
ユーザ	http://サーバ名:ポート/imart/ログイングループ ID.portal (ブラウザ) http://サーバ名:ポート/imart ログイングループ ID.mobile (i-mode) 標準では、上記 2 つの URL を用意しています。

4.2 ログインシーケンス

認証処理は、以下のシーケンスによって構成されています。

図中のモジュール部分は、設定ファイルによって、処理を変更できるモジュールとなっています。

- リクエストから渡される引数の解析方法を変更可能です。
- 認証の方法を変更することが可能です。

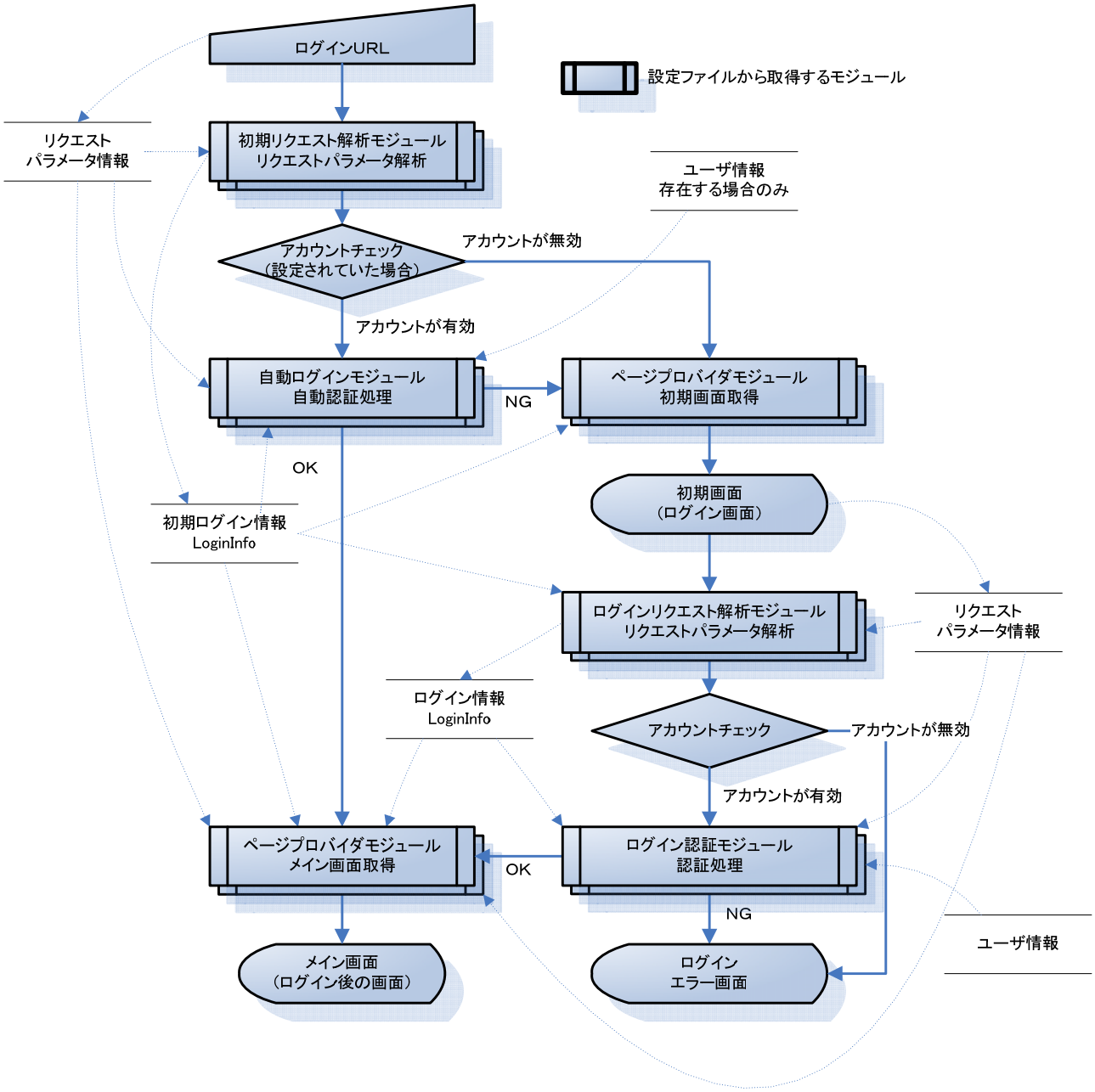


図 47 ログインシーケンス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.2.1 ログインシーケンスの解説

ログインシーケンス上には取り替え可能なモジュールが存在します。
ここでは、各処理とモジュールの動作の概要について間単に解説します。

■ 初期リクエスト解析モジュール

リクエストパラメータ情報より有効なパラメータをを取り出して、ログイン情報に保管します。
このモジュールは、複数設定可能です。
複数設定した場合は、設定された順序ですべてのモジュールを実行します。

■ 自動ログインモジュール

ログイン情報を用いて、認証処理を行いその結果を返します。
このモジュールは、複数設定可能です。また、設定しなくてもかまいません。
設定しないことで、自動ログインを無効にできます。
複数設定した場合は、設定された順序で有効なモジュールを1つ選択し、実行します。

■ ログインリクエスト解析モジュール

リクエストパラメータ情報より有効なパラメータをを取り出して、ログイン情報に保管します。
（このモジュールに渡されるログイン情報は、初期リクエスト解析モジュール群で保管されたものが引き継がれています。）
このモジュールは、複数設定可能です。
複数設定した場合は、設定された順序ですべてのモジュールを実行します。

■ ログインモジュール

ログイン情報を用いて、認証処理を行いその結果を返します。
このモジュールは、複数設定可能です。最低でも1つは設定が必要です。
複数設定した場合は、設定された順序で有効なモジュールを1つ選択し、実行します。

■ アカウントチェック

アカウントの検証を行います。チェック内容は以下の通りです。

- アカウントの存在チェック
- アカウントの有効期間チェック
- アカウントのライセンスチェック
- アカウントのロックのチェック

■ ページプロバイダモジュール

ログイン画面情報およびメイン画面情報を返します。
このモジュールは、複数設定可能です。
複数設定した場合は、設定された順序でモジュールを実行し、画面情報が取得できたところで、
処理を中断し、その画面情報を利用します。

4.3 ログインセッション情報

ログインを行う場合にログインユーザの情報をセッションに保管しています。

デフォルトのセッションパラメータ名は、jp.co.intra_mart.foundation.security.SecurityConstraint に定数値として定義されています。

表 52 ログインセッション情報

項目	値	セッションパラメータ名（デフォルト）
ユーザ ID	ログインユーザID	jp.co.intra_mart.session.user
ログイングループ	ログイングループ ID	jp.co.intra_mart.session.group
ログインタイプ	super (システム管理者) group (アカウント) user (ユーザ)	jp.co.intra_mart.session.login_type
カラーパターン ID	カラーパターン ID	jp.co.intra_mart.session.color_pattern_id
ロケール	ログインロケール	jp.co.intra_mart.session.locale
エンコーディング	ログインエンコーディング	jp.co.intra_mart.session.encoding
クライアントタイプ	pc mobile	jp.co.intra_mart.session.client_type
ログイン時刻	ログインした時間 (YYYY/MM/DD|HH:MI:SS)	jp.co.intra_mart.session.login_time
ログイン署名	ログイン署名	jp.co.intra_mart.session.signature
ログイン URL	ログイン画面の URL	jp.co.intra_mart.session.initial.url

4.3.1 セッション情報インタフェース

パッケージは、jp.co.intra_mart.foundation.security.certification です。

■ **SessionInfo インタフェース**

ログインセッション情報を参照するためのインタフェース。

また、任意のセッション属性についてもアクセスできます。

ログインセッションの情報は必ずこのインタフェースを通して参照します。

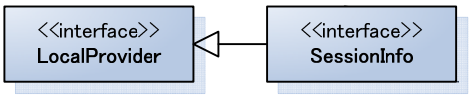


図 48 SessionInfo インタフェース

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.3.2 ログインセッション情報設定の定義

ログインセッション情報設定ではログインセッションに保管する情報のパラメータ名を任意に変更できます。

変更したい項目だけ記述します。記述しない項目は、デフォルトのパラメータ名が使用されます。

通常、変更の必要はありません。

session-config での例を以下に示します。

```
<access-security>
  <session-config>
    <login-type-name>my_login_type</login-type-name>
    <client-type-name>my_client_type</client-type-name>
    <locale-name>my_locale</locale-name>
    <encoding-name>my_encoding</encoding-name>
    <login-group-name>my_login_group</login-group-name>
    <user-name>my_user</user-name>
    <login-time-name>my_login_time</login-time-name>
    <signature-name>my_signature</signature-name>
    <color-pattern-id-name>my_color_pattern_id</color-pattern-id-name>
  </session-config>
</access-security>
```

図 49 ログインセッション情報設定例

■ タグの説明

タグ名	説明
login-type-name	ログインタイプをセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。
client-type-name	クライアントタイプをセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。
locale-name	ロケールをセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。
encoding-name	エンコーディングをセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。
login-group-name	ログイングループをセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。
user-name	ユーザIDをセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。
login-time-name	ログイン時間をセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。
signature-name	ログイン署名をセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。
color-pattern-id-name	カラーパターン ID をセッションに保管するためのパラメータ名を設定します。 省略した場合は、「表 52 ログインセッション情報」の値となります。

4.4 ログインリクエスト情報

初期画面、ログイン画面からのリクエストパラメータ(URL 引数)は以下のものがあります。(デフォルト)
標準では、このパラメータ名でリクエストを行うことで、ログインユーザ情報としてセッションに書き込まれます。
このパラメータをを処理するのは、初期およびログインリクエスト解析モジュールです。

表 53 ログインリクエストパラメーター覧

パラメータ	説明	パラメータ名(デフォルト)
ログインタイプ	ログインのタイプ(参照のみ)	im_login_type
クライアントタイプ	ログインするクライアントのタイプ	im_client_type
ロケール	ロケール	im_locale
エンコーディング	ログインエンコーディング	im_encoding
ログイングループ	ログイングループ	im_group
ユーザID	認証するユーザのID	im_user
パスワード	認証するユーザのパスワード	im_password
ダイレクト URL	認証後表示するメインページ	im_url
ダイレクト URL 署名	ダイレクトURLの署名	im_url_sign

例えば、ユーザID、パスワードをリクエストに指定して、アクセスする場合は、以下のようになります。

```
http://サーバ名:ポート/imart/ログイングループ ID.portal?im_user=ユーザ名&im_password=パスワード
```

4.4.1 ログインリクエスト情報インタフェース

パッケージは、jp.co.intra_mart.foundation.security.certification です。

■ **LoginInfo インタフェース**

ログインに関する情報を取得するインタフェース
ログイン情報は必ずこのインタフェースを通して参照します。

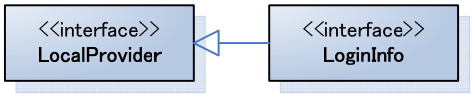


図 50 LoginInfo インタフェース

■ **LoginRequestInfo インタフェース**

ログインに関するリクエストパラメータ操作するインタフェース
ログイン情報は必ずこのインタフェースを通して操作します。

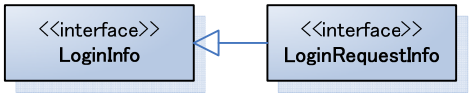


図 51 LoginRequestInfo インタフェース

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.5 ログイン時のロケール

アクセス時のロケール決定は、以下の順序で行います。

■ システム管理者

アクセス状態	ロケール決定順序
初期アクセス時	URLパラメータのロケール(対応範囲)
	システム管理者のロケール
	ブラウザのロケール(対応範囲)
	システムロケール（設定ファイル）
ログイン時	URLパラメータのロケール(対応範囲)
	システム管理者のロケール
	ブラウザのロケール(対応範囲)
	システムロケール（設定ファイル）

■ グループ管理者

アクセス状態	ロケール決定順序
初期アクセス時	URLパラメータのロケール(対応範囲)
	グループ管理者のロケール
	ログイングループのロケール
	ブラウザのロケール(対応範囲)
	システムロケール（設定ファイル）
ログイン時	URLパラメータのロケール(対応範囲)
	グループ管理者のロケール
	ログイングループのロケール
	ブラウザのロケール(対応範囲)
	システムロケール（設定ファイル）

■ ユーザ

アクセス状態	ロケール決定順序
初期アクセス時	URLパラメータのロケール(対応範囲)
	ログイングループのロケール
	ブラウザのロケール(対応範囲)
	システムロケール（設定ファイル）
ログイン時	URLパラメータのロケール(対応範囲)
	ユーザのロケール
	ログイングループのロケール
	ブラウザのロケール(対応範囲)
	システムロケール（設定ファイル）

4.6 ログイン時のカラーパターン

アクセス時のカラーパターン決定は、以下の順序で行います。

■ システム管理者

アクセス状態	カラーパターン決定順序
初期アクセス時	URLパラメータのカラーパターン(対応範囲)
	システム管理者のカラーパターン
	システムカラーパターン（設定ファイル）
ログイン時	URLパラメータのカラーパターン(対応範囲)
	システム管理者のカラーパターン
	システムカラーパターン（設定ファイル）

■ グループ管理者

アクセス状態	カラーパターン決定順序
初期アクセス時	URLパラメータのカラーパターン(対応範囲)
	グループ管理者のカラーパターン
	ログイングループのカラーパターン
	システムカラーパターン（設定ファイル）
ログイン時	URLパラメータのカラーパターン(対応範囲)
	グループ管理者のカラーパターン
	ログイングループのカラーパターン
	システムカラーパターン（設定ファイル）

■ ユーザ

アクセス状態	カラーパターン決定順序
初期アクセス時	URLパラメータのカラーパターン(対応範囲)
	ログイングループのカラーパターン
	システムカラーパターン（設定ファイル）
ログイン時	URLパラメータのカラーパターン(対応範囲)
	ユーザのカラーパターン
	ログイングループのカラーパターン
	システムカラーパターン（設定ファイル）

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7 認証処理の設定

認証処理では、ログインシーケンスにあった通りいくつかのモジュール組み合わせることで処理を行っています。
ここでは、認証処理における各モジュールの説明のその設定方法について解説します。
認証処理は各ユーザ（システム管理者、グループ管理者、ユーザ）個別に設定を行います。
設定ファイルは、サーバマネージャ/conf/access-security.xml で設定を行います。
各ユーザ別の設定用のタグは以下の通りです。

表 54 認証処理モジュール設定を行うタグ一覧

ユーザの種類	対象のタグ名
システム管理者	<security-config>の<super-user-security>
グループ管理者	<security-config>の<group-super-user-security>
ユーザ	<security-config>の<user-security>

4.7.1 モジュール初期化インタフェース

認証処理に用いる各モジュールは、以下のインタフェースを実装することで、インスタンス化された時に初期化パラメータを引数にした初期化用のメソッドが実行されるようになります。

- **jp.co.intra_mart.foundation.security.XmlInitParamable インタフェース**
  XML 形式の初期化処理インタフェース。
  実装クラス名が設定されているタグの親タグの Node を引数に渡して初期化メソッドを呼び出します。

- **jp.co.intra_mart.foundation.security.PropertyInitParamable インタフェース**
  プロパティ形式の初期化処理インタフェース。
  実装クラス名が設定されているタグの親タグ内のすべての<init-param>タグを探し出し、
  Map 形式でキー（パラメータ名<param-name>）,値（パラメータ値<param-value>）を保管したものを引数に渡して初期化メソッドを呼び出します。

4.7.2 リクエスト解析モジュール

初期画面（ログイン画面）の要求リクエスト、およびログイン画面からの認証リクエストからログインリクエスト情報を作成するクラスを定義します。

リクエスト解析モジュールは、複数設定可能です。
複数設定した場合は、設定された順序ですべてのモジュールを実行します。

4.7.2.1 リクエスト解析モジュールインタフェース

パッケージは、jp.co.intra_mart.foundation.security.certification です。

- **RequestAnalyzer インタフェース**
  リクエスト解析モジュールはこのインタフェースを実装する必要があります。
  リクエストを解析し、リクエスト情報を返却するメソッドを持ちます。
- **ExtendsRequestAnalyzer インタフェース**
  リクエスト解析モジュールはこのインタフェースを実装する必要があります。
  リクエストを解析し、リクエスト情報を返却するメソッドを持ちます。
  Ver.7.0 よりカスタマイズしやすいインタフェースになっています。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7.2.2 リクエスト解析モジュール設定の定義

<super-user-security>、<group-super-user-security>、<user-security>では<initial-request-analyzer>と<login-request-analyzer>の2カ所に存在します。

<initial-request-analyzer>では、初期画面要求時（ログイン画面要求）のリクエストパラメータを解析します。

<login-request-analyzer>では、ログイン認証時のリクエストパラメータを解析します。

user-security での例を以下に示します。

```
<access-security>
  <security-config>
    <user-security>

      <initial-request-analyzer>
        <request-analyzer-class>SampleRequestAnalzer2</request-analyzer-class>
      </initial-request-analyzer>

      <initial-request-analyzer>
        <request-analyzer-class>SampleRequestAnalzer1</request-analyzer-class>
      </initial-request-analyzer>

      <login-request-analyzer>
        <request-analyzer-class>SampleRequestAnalzer1</request-analyzer-class>
      </login-request-analyzer>

      <login-request-analyzer>
        <request-analyzer-class>SampleRequestAnalzer2</request-analyzer-class>
        <init-param>
          <param-name>text</param-name>
          <param-value>xxxx</param-value>
        </init-param>

        .........

      </login-request-analyzer>

      .........

    </user-security>
  </security-config>
</access-security>
```

■ タグの説明

<initial-request-analyzer>		初期画面要求時のリクエストパラメータを解析するモジュールを設定します。 （複数設定可能） <super-user-security>、<group-super-user-security>、<user-security>タグに存在します。 複数設定した場合は、設定した順序で、すべてのモジュールを実行します。
	<request-analyzer-class>	初期画面要求時のリクエストパラメータを解析するクラスを設定します。 RequestAnalyzser または ExtendsRequestAnalyzser インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。 ExtendsRequestAnalyzser ではモジュール初期化インタフェースを実装する必要があります。

<login-request-analyzer>		ログイン認証要求時のリクエストパラメータを解析するモジュールを設定します。 （複数指定可能） <super-user-security>、<group-super-user-security>、<user-security>タグに存在します。 複数設定した場合は、設定した順序で、すべてのモジュールを実行します。
	<request-analyzer-class>	ログイン認証面要求時のリクエストパラメータを解析するクラスを設定します。 RequestAnalyzser または ExtendsRequestAnalyzser インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。 (複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。 ExtendsRequestAnalyzser ではモジュール初期化インタフェースを実装する必要があります。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7.2.3 リクエスト解析モジュール実装クラス

■ **jp.co.intra_mart.foundation.security.certification.StandardRequestAnalyzerクラス**

標準のログインリクエスト解析クラス。

各リクエストパラメータから情報を取得してLoginInfoに設定します。

各リクエストパラメータから情報が取得されなかったものは、LoginInfoに設定しません。

解析では、デフォルトのリクエストパラメータ「表 53 ログインリクエストパラメータ一覧」を使用して各情報を取得します。

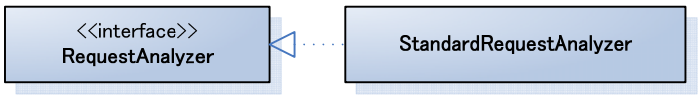


図 52 StandardRequestAnalyzer クラス

◆ StandardRequestAnalyzerクラスのパラメータ

StandardRequestAnalyzerクラスでは、各リクエストパラメータ名を変更できます。

パラメータを指定しない場合は「表 53 ログインリクエストパラメータ一覧」の値が使用されます。

表 55 StandardRequestAnalyzer クラスのパラメータの一覧

パラメータ名	説明
login-type	ログインタイプが設定されるパラメータ名を設定。
client-type	クライアントタイプが設定されるパラメータ名を設定。
locale	ロケールが設定されるパラメータ名を設定。
encoding	エンコーディングが設定されるパラメータ名を設定。
login-group	ログイングループが設定されるパラメータ名を設定。
user	ユーザ ID が設定されるパラメータ名を設定。
password	パスワードが設定されるパラメータ名を設定。

■ **jp.co.intra_mart.foundation.security.certification.StandardLoginRequestAnalyzerクラス**

標準の認証ログインリクエスト解析クラス。

ユーザおよびパスワードのリクエストパラメータから情報を取得してLoginInfoに設定します。各リクエストパラメータから情報が取得されなかったものは、LoginInfoに設定しません。

解析では、デフォルトのリクエストパラメータ「表 53 ログインリクエストパラメータ一覧」を使用して各情報を取得します。

図 53 StandardLoginRequestAnalyzer クラス

◆ StandardLoginRequestAnalyzerクラスのパラメータ

StandardLoginRequestAnalyzerクラスでは、各リクエストパラメータ名を変更できます。

パラメータを指定しない場合は「表 53 ログインリクエストパラメータ一覧」の値が使用されます。

表 56 StandardLoginRequestAnalyzer クラスのパラメータの一覧

パラメータ名	説明
user	ユーザ ID が設定されるパラメータ名を設定。
passward	パスワードが設定されるパラメータ名を設定。

■ **jp.co.intra_mart.foundation.security.certification.SSORequestAnalyzer クラス**

SSO のログインリクエスト解析クラス。

各リクエストパラメータから情報を取得して LoginInfo に設定します。

各リクエストパラメータから情報が取得されなかったものは、LoginInfo に設定しません。

解析では、im-SecureSginOn のクッキー情報からログイン情報を取得します。

m-SecureSginOn のクッキー情報からアカウントの情報が取得できた場合は、認証は完了したとみなします。

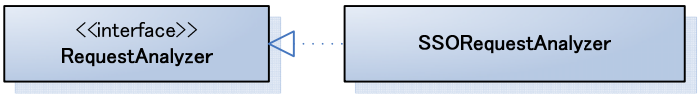


図 54 SSORequestAnalyzer クラス

◆ SSORequestAnalyzer クラスのパラメータ

SSORequestAnalyzer クラスでは、ログアウト時の URL を設定できます。

表 57 SSORequestAnalyzer クラスのパラメータの一覧

パラメータ名	説明
sso-logout-url	ログアウト時に呼び出される URL を設定

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ **jp.co.intra_mart.foundation.security.certification.ShortCutInitialRequestAnalyzer クラス**

ショートカット ID をリクエストパラメータから取得して LoginInfo に設定します。
リクエストパラメータからショートカット ID が取得されなかった場合は LoginInfo に設定しません。
ショートカット ID に紐づくショートカット情報に、認証が不要の設定がされていた場合、認証は完了したとみなします。
解析では、デフォルトのリクエストパラメータ名を使用して情報を取得します。

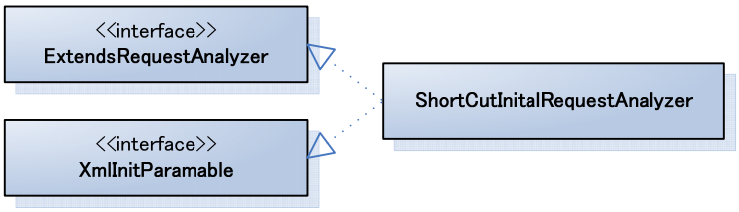


図 55 ShortCutInitialRequestAnalyzer クラス

◆ ShortCutInitialRequestAnalyzer クラスのパラメータ

ShortCutInitialRequestAnalyzer クラスでは、各リクエストパラメータ名を変更できます。
パラメータを設定しない場合は、デフォルト値が使用されます。

表 58 StandardLoginRequestAnalyzer クラスのパラメータの一覧

パラメータ名	説明	デフォルト値
shortcut	ショートカットIDが設定されるパラメータ名を設定します。	im_shortcut

■ **jp.co.intra_mart.foundation.security.certification.DirectUrlInitialRequestAnalyzer クラス**ダイレクト URL ログインリクエスト解析クラス。

ダイレクト URL とダイレクト URL 署名をリクエストパラメータから取得して LoginInfo に設定します。
リクエストパラメータから情報が取得されなかった場合は LoginInfo に設定しません。
解析では、デフォルトのリクエストパラメータ名を使用して情報を取得します。

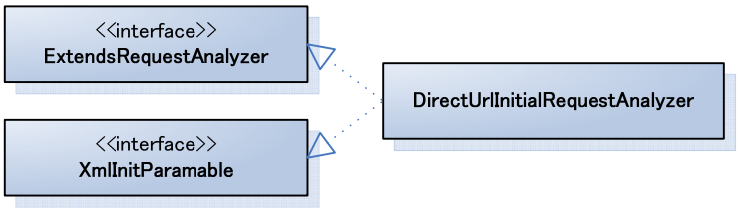


図 56 ShortCutInitialRequestAnalyzer クラス

◆ DirectUrlInitialRequestAnalyzer クラスのパラメータ

DirectUrlInitialRequestAnalyzer クラスでは、各リクエストパラメータ名を変更できます。
パラメータを設定しない場合は、デフォルト値が使用されます。

表 59 StandardLoginRequestAnalyzer クラスのパラメータの一覧

パラメータ名	説明	デフォルト値
url	ダイレクトURLが設定されるパラメータ名を設定します。	im_url
url-sign	ダイレクトURL署名が設定されるパラメータ名を設定します。	im_url_sign

■ **jp.co.intra_mart.foundation.security.certification.MobileAgentInitialRequestAnalyzer クラス**

かんたんログイン用のログインリクエスト解析クラス。

まず、クライアントから送信されたモバイル端末個体識別番号を取得し、該当するアカウントを特定します。次に、そのアカウントの属性情報、モバイルパスワード、モバイル端末個体識別番号を元にアクセスキーを算出します。（参照：API リスト「jp.co.intra_mart.foundation.security.mobile.MobileEasyLoginUtil」クラス）

最後に、算出したアクセスキーと、クライアントから送信されたアクセスキーを比較し、合致した場合のみ、認証が完了したとみなします。

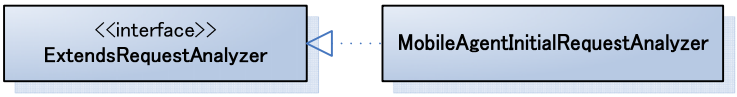


図 57 MobileAgentInitialRequestAnalyzer クラス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7.3 認証モジュール

リクエスト情報を元に認証を行うクラスを定義します。

認証モジュールは複数設定可能です。
複数設定されている場合は、設定されている順序で有効な認証モジュールかどうか検証し、最初に有効である認証モジュールを使用して、認証処理を行います。
（複数のモジュールの中から 1 つのモジュールを選択します。）

また、認証モジュールに認証リスナーが存在する場合、認証後に、認証リスナーが実行されます。
認証リスナーは、認証モジュールに対して、複数設定可能です。

4.7.3.1 認証モジュールインタフェース

パッケージは、jp.co.intra_mart.foundation.security.certification です。

- **SuperCertification インタフェース**
  システム管理者およびグループ管理者の認証モジュールはこのインタフェースを実装する必要があります。
  <super-user-security>、<group-super-user-security>で利用します。
  リクエスト情報と管理者情報から、認証を行うメソッドを持ちます。
  認証結果は、CR_OK および CR_NG です。

- **ExtendsSuperCertification インタフェース**
  システム管理者およびグループ管理者の認証モジュールはこのインタフェースを実装する必要があります。
  <super-user-security>、<group-super-user-security>で利用します。
  リクエスト情報と管理者情報から、認証を行うメソッドを持ちます。
  認証結果は、CR_OK および CR_NG です。
  Ver.7.0 よりカスタマイズしやすいインタフェースになっています。

- **UserCertification インタフェース**
  ユーザの認証モジュールはこのインタフェースを実装する必要があります。
  <user-security>で利用します。
  リクエスト情報とユーザ情報から、認証を行うメソッドを持ちます。
  認証結果は、CR_OK および CR_NG です。

- **ExtendsUserCertification インタフェース**
  ユーザの認証モジュールはこのインタフェースを実装する必要があります。
  <user-security>で利用します。
  リクエスト情報とユーザ情報から、認証を行うメソッドを持ちます。
  認証結果は、CR_OK および CR_NG です。
  Ver.7.0 よりカスタマイズしやすいインタフェースになっています。

4.7.3.2 認証条件インタフェース

認証を行う条件を付加するためのインタフェースです。
認証モジュールを条件により選択する場合は認証モジュールインタフェースと共にこのインターフェースを実装します。このインタフェースの条件(validate メソッド)が true を返した場合は、このモジュールが有効であると判断します。
認証モジュールが認証条件インタフェースを実装しない場合は、常に有効な認証モジュールとして扱われます。

パッケージは、jp.co.intra_mart.foundation.security.certification です。

- **SuperCertificationValidation インタフェース**
  システム管理者およびグループ管理者の条件付きの認証モジュールはこのインタフェースを実装する必要があります。
  <super-user-security>、<group-super-user-security>で利用します。

- **UserCertificationValidation インタフェース**
  ユーザの条件付きの認証モジュールはこのインタフェースを実装する必要があります。
  <user-security>で利用します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7.3.3 認証リスナーインタフェース

パッケージは、jp.co.intra_mart.foundation.security.certification です。

- **SuperCertificationListener インタフェース**

  システム管理者およびグループ管理者の認証リスナーはこのインタフェースを実装する必要があります。
  <super-user-security>、<group-super-user-security>の<auto-certification>と<certification>で利用します。
  ログイン認証後に認証結果、ログイン情報、管理者情報、リクエスト情報およびレスポンス情報から実行されるリスナーメソッドを持ちます。

- **ExtendsSuperCertificationListener インタフェース**

  システム管理者およびグループ管理者の認証リスナーはこのインタフェースを実装する必要があります。
  <super-user-security>、<group-super-user-security>の<auto-certification>と<certification>で利用します。
  ログイン認証後に認証結果、ログイン情報、管理者情報、リクエスト情報およびレスポンス情報から実行されるリスナーメソッドを持ちます。
  Ver.7.0 よりカスタマイズしやすいインタフェースになっています。

- **UserCertificationListener インタフェース**

  ユーザの認証リスナーはこのインタフェースを実装する必要があります。
  <user-security>の<auto-certification>と<certification>で利用します。
  ログイン認証後に認証結果、ログイン情報、管理者情報、リクエスト情報およびレスポンス情報から実行されるリスナーメソッドを持ちます。

- **ExtendsUserCertificationListener インタフェース**

  ユーザの認証リスナーはこのインタフェースを実装する必要があります。
  <user-security>の<auto-certification>と<certification>で利用します。
  ログイン認証後に認証結果、ログイン情報、管理者情報、リクエスト情報およびレスポンス情報から実行されるリスナーメソッドを持ちます。
  Ver.7.0 よりカスタマイズしやすいインタフェースになっています。

4.7.3.4 認証モジュール設定の定義

<super-user-security>、<group-super-user-security>、<user-security>では<auto-certification>と<certification>の2カ所に存在します。

<auto-certification>では、初期画面要求時のリクエストに対して自動ログイン認証を試みます。

自動ログインの必要がない場合、<auto-certification>のタグは不要です。

<certification>では、ログイン認証時のリクエストに対してログイン認証を試みます。

user-security での例を以下に示します。

```
<access-security>
  <security-config>
    <user-security>

      .........

      <auto-certification>
        <certification-class>SampleStandardUserCertification1</certification-class>
      </auto-certification>

      <auto-certification>
        <certification-class>SampleStandardUserCertification2</certification-class>
      </auto-certification>

      <certification>
        <certification-class>SampleStandardUserCertification1</certification-class>
      </certification>

      <certification>
        <certification-class>SampleStandardUserCertification2</certification-class>
         <init-param>
           <param-name>address-filter</param-name>
           <param-value>192.168.0.*</param-value>
         </init-param>
         <certification-listener>
           <listener-class>SampleUserCertificationListener</listener-class>
         </certification-listener>
         .........
      </certification>

      .........

    </user-security>
  </security-config>
</access-security>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ タグ説明

<auto-certification>			初期画面要求時に自動ログイン認証を行うモジュールを設定します。 （複数設定可能） 初期画面要求時に自動ログインを行わないようにする場合は、 このタグは必要ありません。 <super-user-security>、<group-super-user-security>、<user-security>タグに 存在します
	<certification-class>		初期画面要求時に自動ログイン認証を行うクラスを設定します。 <super-user-security>、<group-super-user-security>タグ内では SuperCertification または ExtendsSuperCertification インタフェースを実装する必要があります。 <user-security>タグ内では UserCertification または ExtendsUserCertification インタフェースを実装する必要があります。 認証モジュール設定<auto-certification>は複数設定することが可能ですが、どのモジュールが有効か判定する認証条件インタフェースを実装した、認証モジュールが必要です。
	<init-param>		このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。 ExtendsSuperCertification または ExtendsUserCertification ではモジュール初期化インタフェースを実装する必要があります。
	<certification-listener>		自動ログイン認証後に処理を行うためのリスナーを設定します。 (複数指定可能) 認証後に何も処理する必要がない場合は、このタグは必要ありません。
		<listener-class>	自動ログイン認証後に処理を行うためのリスナークラスを設定します。 <super-user-security>、<group-super-user-security>タグ名では SuperCertificationListener インタフェースを実装する必要があります。 <user-security>タグ内では UserCertificationListener または ExtendsUserCertificationListener インタフェースを実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。 ExtendsSuperCertificationListener または ExtendsUserCertificationListener ではモジュール初期化インタフェースを実装する必要があります。

<certification>			ログイン認証要求時にログイン認証を行うモジュールを設定します。 （複数設定可能） 初期画面要求時に自動ログインを行わないようにする場合は、 このタグは必要ありません。 <super-user-security>、<group-super-user-security>、<user-security>タグに 存在します
	<certification-class>		ログイン認証要求時にログイン認証を行うクラスを設定します。 <super-user-security>、<group-super-user-security>タグ内では SuperCertification または ExtendsSuperCertification インタフェースを実装する必要があります。 <user-security>タグ内では UserCertification または ExtendsUserCertification インタフェースを実装する必要があります。 認証モジュール設定<certification>は複数設定することが可能ですが、どのモジュールが有効か判定する認証条件インタフェースを実装した、認証モジュールが必要です。
	<init-param>		このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。 ExtendsSuperCertification または ExtendsUserCertification ではモジュール初期化インタフェースを実装する必要があります。
	<certification-listener>		ログイン認証後に処理を行うためのリスナーを設定します。 (複数指定可能) 認証後に何も処理する必要がない場合は、このタグは必要ありません。
		<listener-class>	ログイン認証後に処理を行うためのリスナークラスを設定します。 <super-user-security>、<group-super-user-security>タグ名では SuperCertificationListener インタフェースを実装する必要があります。 <user-security>タグ内では UserCertificationListener インタフェースを実装する必要があります。
		<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。 ExtendsSuperCertificationListener または ExtendsUserCertificationListener ではモジュール初期化インタフェースを実装する必要があります。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7.3.5 認証モジュール実装クラス

■ **jp.co.intra_mart.foundation.security.certification.StandardSuperUserCertification クラス**

標準の管理者用認証クラス。

メソッド引数 LoginInfo のパスワードと管理者情報のパスワードをチェックして認証を行います。

初期化パラメータで、アクセスできるクライアントの範囲をフィルタリング可能です。

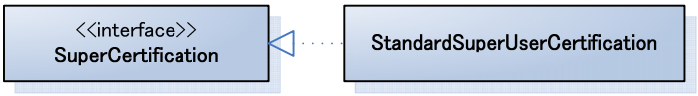


図 58 StandardSuperUserCertification クラス

◆ StandardSuperUserCertification クラスのパラメータ

表 60 StandardSuperUserCertification クラスのパラメータの一覧

パラメータ名	説明
address-filter	クライアントの IP アドレスでフィルタリングします。 IP アドレスの正規表現を設定。 この設定に一致しないクライアントは、認証されません。

■ **jp.co.intra_mart.foundation.security.certification.AdminUserCertification クラス**

ログイングループ管理権限ユーザ用認証クラス。

ログイングループ管理権限を委譲されたユーザのパスワードとユーザ情報のパスワードをチェックして認証を行います。

この認証モジュールが使用される条件は、ログイングループ管理権限ユーザ用モードでのログイン処理の時のみです。

<group-super-security>の<certification>で設定するクラス

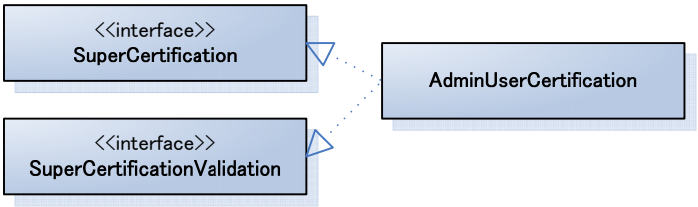


図 59 AdminUserCertification クラス

■ **jp.co.intra_mart.foundation.security.certification.StandardUserCertification クラス**

標準のユーザ用認証クラス。

LoginInfo のパスワードとユーザ情報のパスワードをチェックして認証を行います。

モバイル用のパスワードを利用するクライアントタイプをパラメータで指定できます。

クライアントタイプがモバイル用のパスワードを利用するクライアントタイプの場合は、LoginInfo のパスワードとユーザ情報のモバイルパスワードをチェックして認証を行います。

図 60 StandardUserCertification クラス

◆ StandardUserCertification クラスのパラメータ

表 61 StandardUserCertification クラスのパラメータの一覧

パラメータ名	説明
mobile-client-type	モバイルパスワードで認証を行うクライアントタイプを設定。 （複数設定可能）。 例）pda,mobile

■ **jp.co.intra_mart.foundation.security.certification.LDAPUserCertification クラス**

LDAP でのユーザ用認証クラス。

LoginInfo のパスワードとユーザ ID で LDAP に対して認証を行います。

LDAP 情報はログイングループ単位でパラメータとして設定します。

LDAP 情報が設定されていないログイングループでは、通常の認証を行います。

パラメータで DN（識別情報）の設定は、ユーザ ID の部分を‘？’で記述します。

uid=?,ou=People, o=ldaps.intra-mart.jp

‘?’をログインするユーザ名に置換して LDAP へ認証を依頼します。

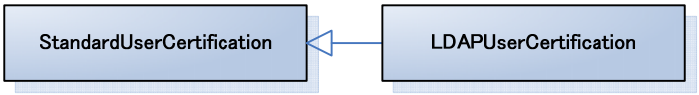


図 61 LDAPUserCertification クラス

◆ LDAPUserCertification クラスのパラメータ

LDAP への接続情報の設定を行います。

ログイングループ単位で設定します。

表 62 LDAPUserCertification クラスの必須パラメータの一覧

必須パラメータ名	説明
ログイングループ ID.provider-url	プロバイダ URL を設定します。(Ex : ldap://localhost:389/) カンマ(,)区切りで複数の URL を指定できます。 (Ex : ldap://localhost:389/, ldap://localhost:27272/)。 複数指定した場合、左から順に LDAP サーバに通信を試みます。通信できない場合のみ次の接続先に通信を試みます。
ログイングループ ID.dn	識別情報を設定します。 （Ex : Xuid=?,ou=People, o=ldaps.intra-mart.jp）
ログイングループ ID.context-factory	コンテキストファクトリを設定します。 （Ex : com.sun.jndi.ldap.LdapCtxFactory）

表 63 LDAPUserCertification クラスの任意パラメータの一覧

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

任意パラメータ名	説明
ログイングループ ID. load-balancing	ロードバランサを有効にします。(true/false) 設定しない場合は false として扱います。 false を指定した場合、プロバイダ URL を左から順に認証を試みます。 true を指定した場合、プロバイダ URL の開始位置をランダムに決定し、決定した位置を基準に右方向に順に認証を試みます。
ログイングループ ID. connect-timeout-property-name	接続タイムアウトを設定するプロパティ名を設定します。 （Ex : com.sun.jndi.ldap.connect.timeout）
ログイングループ ID. connect-timeout	接続タイムアウト時間を設定します。 プロパティに com.sun.jndi.ldap.connect.timeout を指定した場合、ミリ秒数を示す 0 より大きい整数で指定し、LDAP プロバイダがこの値の期間中に接続を確立できないと、接続の試行は中止されます。 0 以下の整数指定した場合または設定しない場合は、TCP などのネットワークプロトコルのタイムアウト値が使用されます。
ログイングループ ID. attempt-on-failed-authentication	LDAP 認証失敗(ユーザ ID またはパスワードが不正)時に次のプロバイダ URL に認証を試みます(true/false)。 設定しない場合は false として扱います。 false を指定した場合、次のプロバイダ URL に認証を試みません。 true を指定した場合、次のプロバイダ URL に認証を試みます。
permit-no-password	空パスワードによる認証を許可するかどうかを設定します。(true/false) 設定しない場合は false として扱います。 false を指定した場合、空パスワードを認証失敗とみなします。 true を指定した場合、空パスワードで LDAP に問い合わせます。
log	ログをコンソールに出力かどうかを設定します。(true/false) 設定しない場合は false として扱います。 デバッグ用のオプションです。

■ **jp.co.intra_mart.foundation.security.certification.ldap.extension.LDAPUserCertification クラス**

複数 OU 対応版の LDAP でのユーザ用認証クラス。
LoginInfo のパスワードとユーザ ID で LDAP に対して認証を行います。
LDAP 情報はログイングループ単位でパラメータとして設定します。
LDAP 情報が設定されていないログイングループでは、通常の認証を行います。
ログイングループ ID.base-dn を元に、ログイングループ ID.serch-filter の'?'の部分をユーザ ID に置換して、LDAP 検索を行い、ユーザのフル DN を作成し、そのフル DN で認証を行います。

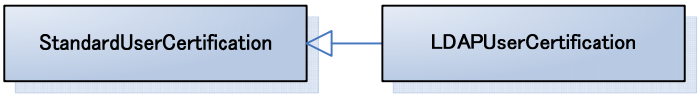


図 62 LDAPUserCertification クラス

◆ LDAPUserCertification クラスのパラメータ

LDAP への接続情報の設定を行います。
ログイングループ単位で設定します。

表 64 LDAPUserCertification クラスの必須パラメータの一覧

必須パラメータ名	説明
ログイングループ ID.provider-url	プロバイダ URL を設定します。 (Ex : ldap://localhost:389/) カンマ(,)区切りで複数の URL を指定できます。 (Ex : ldap://localhost:389/, ldap://localhost:27272/)。 複数指定した場合、左から順に LDAP サーバに通信を試みます。通信できない場合のみ次の接続先に通信を試みます。
ログイングループ ID.context-factory	コンテキストファクトリを設定します。 （Ex : com.sun.jndi.ldap.LdapCtxFactory）
ログイングループ ID.base-dn	識別情報のベースを設定します。 （Ex : o=ldaps.intra-mart.jp）
ログイングループ ID.serch-filter	ユーザを特定する為に一意になる属性を設定します。 （Ex : uid=?）

表 65 LDAPUserCertification クラスの検索パラメータの一覧

検索パラメータ名	説明
ログイングループ ID.scope	検索レベルを設定します。 SUBTREE_SCOPE ベースDNで指定したPath配下のサブツリーすべてを検索
ログイングループ ID.count-limit	最大検索数を設定します。デフォルト:0（上限無し）
ログイングループ ID.time-limit	時間制限（ミリ秒）を設定します。デフォルト:0（無制限）
ログイングループ ID.deref-linf	間接参照の有無を設定します。デフォルト:false
ログイングループ ID.returning-obj	結果の一部がオブジェクトとして返るかを設定します。 デフォルト:false
ログイングループ ID.returning-attribute	結果として返される属性を設定します。 デフォルト:指定なし(すべての属性) 属性 ID 　指定した属性のみ返す。 　（複数指定可能:[,]カンマで区切る） 指定なし 　すべての属性を返す

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

表 66 LDAPUserCertification クラスの任意パラメータの一覧

任意パラメータ名	説明
ログイングループ ID.load-balancing	ロードバランサを有効にします。(true/false) 設定しない場合は false として扱います。 false を指定した場合、プロバイダ URL を左から順に認証を試みます。 true を指定した場合、プロバイダ URL の開始位置をランダムに決定し、決定した位置を基準に右方向に順に認証を試みます。
ログイングループ ID.connect-timeout-property-name	接続タイムアウトを設定するプロパティ名を設定します。 （Ex : com.sun.jndi.ldap.connect.timeout）
ログイングループ ID.connect-timeout	接続タイムアウト時間を設定します。 プロパティに com.sun.jndi.ldap.connect.timeout を指定した場合、ミリ秒数を示す 0 より大きい整数で指定し、LDAP プロバイダがこの値の期間中に接続を確立できないと、接続の試行は中止されます。 0 以下の整数指定した場合または設定しない場合は、TCP などのネットワークプロトコルのタイムアウト値が使用されます。
ログイングループ ID.attempt-on-failed-authentication	LDAP 認証失敗(ユーザ ID またはパスワードが不正)時に次のプロバイダ URL に認証を試みます(true/false)。 設定しない場合は false として扱います。 false を指定した場合、次のプロバイダ URL に認証を試みません。 true を指定した場合、次のプロバイダ URL に認証を試みます。
ログイングループ ID.serching-dn	LDAP 上でユーザを検索する為のユーザ(DN)を設定します。 (Ex : uid=admin,cn=User,o=ldaps.intra-mart.jp) この設定は、ActiveDirectory の場合には必要になります。
ログイングループ ID.serching-pw	LDAP 上でユーザを検索する為のユーザパスワードを設定します。（Ex : admin） この設定は、ActiveDirectory の場合には必要になります。

表 67 ログイングループ全体に対して設定するパラメータ

任意パラメータ名	説明
permit-no-password	空パスワードによる認証を許可するかどうかを設定します。(true/false) 設定しない場合は false として扱います。 false を指定した場合、空パスワードを認証失敗とみなします。 true を指定した場合、空パスワードで LDAP に問い合わせます。
log	ログをコンソールに出力かどうかを設定します。(true/false) 設定しない場合は false として扱います。 デバッグ用のオプションです。

4.7.3.6 認証リスナー実装クラス

■ **jp.co.intra_mart.foundation.security.activesession.GroupActiveSessionRegister クラス**

グループ管理者がログイン成功時にアクティブセッションを登録するための認証リスナークラス。
メソッド引数の認証結果をチェックして認証が成功していればアクティブセッション情報を登録します。

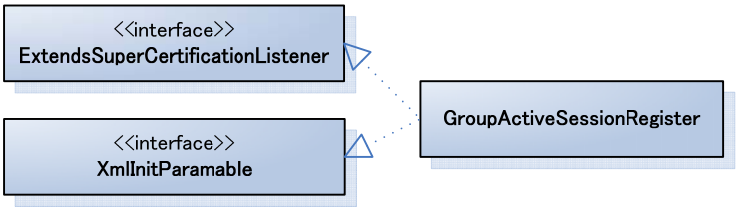


図 63 GroupActiveSessionRegister クラス

◆ GroupActiveSessionRegister クラスのパラメータ

本クラスにパラメータはありません。

◆ 注意事項

GroupActiveSessionRegister を<group-super-user-security>の<certification>の<certification-listener>に必ず設定してください。

■ **jp.co.intra_mart.foundation.security.activesession.UserActiveSessionRegister クラス**

一般ユーザがログイン成功時にアクティブセッションを登録するための認証リスナークラス。
メソッド引数の認証結果をチェックして認証が成功していればアクティブセッション情報を登録します。

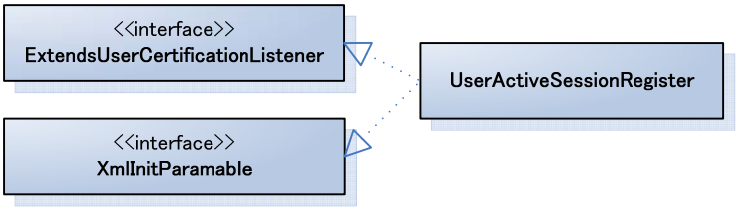


図 64 StandardUserCertification クラス

◆ UserActiveSessionRegister クラスのパラメータ

本クラスにパラメータはありません。

◆ 注意事項

UserActiveSessionRegister を<user-security>の<certification>の<certification-listener>に必ず設定してください。
<user-security>の<auto-certification>が定義されている場合、同様に UserActiveSessionRegister を設定してください。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7.4 ページプロバイダモジュール

ログインに関わる各種ページ URL を返却するクラスを定義します。

- エラーページを返却するページプロバイダモジュールと、各ユーザ別に初期画面（ログイン画面）、標準のメイン画面（ログイン後の画面）を返却するページプロバイダモジュールがあります。
- エラーページを返却するページプロバイダモジュール
- <error-page-provider >の設定では ErrorPageProvider インタフェースを実装する必要があります。
- 初期画面（ログイン画面）、標準のメイン画面（ログイン後の画面）を返却するページプロバイダモジュール
- <super-user-security>の設定では UserPageProvider インタフェースを実装する必要があります。
- <group-super-user-security>の設定では UserPageProvider インタフェースを実装する必要があります。
- <user-security>の設定では UserPageProvider インタフェースを実装する必要があります。
  <user-security>の設定で行う標準のメイン画面の設定は、ログインユーザがメインページパターンを選択していない場合に使用されます。
- このモジュールは、複数設定可能です。
  複数設定した場合は、設定された順序でモジュールを実行し、画面情報が取得できたところで、処理を中断し、その画面情報を利用します。

4.7.4.1 ページプロバイダモジュールが提供するURL

ErrorPageProvider、UserPageProvider が提供する各ページの URL の表現について、以下のルールを定める。

- **スクリプト開発モデルの URL**

  スクリプト開発モデルのパス.jssp を指定

  例） test/sequrity/login/login_01.jssp

- **JAVAEE 開発モデルの URL（JSP、サーブレットといった一般的なページ）**

  コンテキストパス以下の URL を指定

  例） test/sequrity/login/login_01.jsp
  例） test/sequrity/login/login_01.html
  例） test/sequrity/login/LoginServlet

- **外部の URL**

  HTTP://・・・・を指定

  例） http://www.yahoo.co.jp

4.7.4.2 ページプロバイダモジュールインタフェースとモデル

パッケージは、jp.co.intra_mart.foundation.security.certification です。

- **PageUrl クラス**

  ページの URL とその引数を保持するモデルクラスです。

- **ErrorPageProvider インタフェース**

  エラーページプロバイダモジュールはこのインタフェースを実装する必要があります。
  <sequrity-config>の<error-page-provider>で設定するクラスのインタフェース
  認証における各エラーページ（PageUrl）を返却します。
  各種エラーページを返却します。

- **ExtendsErrorPageProvider インタフェース**

  エラーページプロバイダモジュールはこのインタフェースを実装する必要があります。
  <sequrity-config>の<error-page-provider>で設定するクラスのインタフェース
  認証における各エラーページ（PageUrl）を返却します。
  各種エラーページを返却します。
  Ver.7.0 よりカスタマイズしやすいインタフェースになっています。

- **UserPageProvider インタフェース**

  ユーザ用のページプロバイダモジュールはこのインタフェースを実装する必要があります。
  <super-user-security>、<group-super-user-security>、<user-security>の<page-provider>で設定するクラスのインタフェース
  初期ページ、メインページ（PageUrl）を返却します。

- **ExtendsUserPageProvider インタフェース**

  ユーザ用のページプロバイダモジュールはこのインタフェースを実装する必要があります。
  <super-user-security>、<group-super-user-security>、<user-security>の<page-provider>で設定するクラスのインタフェース
  初期ページ、メインページ（PageUrl）を返却します。
  Ver.7.0 よりカスタマイズしやすいインタフェースになっています。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7.4.3 モジュール設定の定義

<page-provider>では、認証処理時に要求するページを返却するページプロバイダを設定します。

<error-page-provider >では、認証処理時に要求するエラーページを返却するページプロバイダを設定します。

error-page-provider,user-security での例を以下に示します。

```
<access-security>
  <security-config>

    <error-page-provider>
      <page-provider-class>SampleErrorPageProvider1</page-provider-class>
      .........
    </error-page-provider>

    <error-page-provider>
      <page-provider-class>SampleErrorPageProvider2</page-provider-class>
      .........
    </error-page-provider>

    <user-security>

      .........
      <page-provider>
        <page-provider-class>SampleUserPageProvider1</page-provider-class>
      </page-provider>

      <page-provider>
        <page-provider-class>SampleUserPageProvider2</page-provider-class>
        <init-param>
          <param-name>initial-page.pc</param-name>
          <param-value>security/login.jsp</param-value>
        </init-param>
        .........
      </page-provider>
      .........

    </user-security>

  </security-config>
</access-security>
```

図 65 ページプロバイダモジュール（ユーザ）の設定例

■ タグの説明

<error-page-provider>		アクセスセキュリティにおける各種エラー画面の取得を行うモジュールを設定します。（複数設定可能） <scurity-config>タグ内に存在します。 複数設定した場合は、設定した順序で実行し、ページが取得できたところで、そページを利用します。
	<page-provider-class>	アクセスセキュリティにおける各種エラー画面の取得を行うクラスを設定します。 ErrorPageProvider または ExtendsErrorPageProvider インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。 ExtendsErrorPageProvider ではモジュール初期化インタフェースを実装する必要があります。

<page-provider>		初期画面およびメイン画面の取得を行うモジュールを設定します。 （複数設定可能） <super-user-security>、<group-super-user-security>、<user-security>タグに存在します。 複数設定した場合は、設定した順序で実行し、ページが取得できたところで、そページを利用します。
	<page-provider-class>	ア初期画面およびメイン画面の取得を行うクラスを設定します。 UserPageProvider または ExtendsUserPageProvider インタフェースを実装する必要があります。
	<init-param>	このクラスに渡す初期化パラメータです。(複数指定可能) このクラスが生成された時に一度だけこのパラメータで初期化します。 ExtendsUserPageProvider ではモジュール初期化インタフェースを実装する必要があります。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.7.4.4 ページプロバイダモジュール実装クラス

■ **jp.co.intra_mart.foundation.security.certification.StandardErrorPageProvider クラス**

標準のエラーページプロバイダクラス。

初期化パラメータにより、各ページの URL を返却します。

<security-config>の< error -page-provider>で設定するクラスです。

図 66 StandardErrorPageProvider クラス

◆ 初期化パラメータ

StandardErrorPageProvider クラスでは、返却するページをパラメータとして設定可能です。

表 68 StandardErrorPageProvider クラスのパラメータの一覧

パラメータ名	説明
certification-error-page	認証処理失敗時のデフォルト画面 URL を設定します。
locked-error-page	ユーザがロック状態の場合のデフォルト画面 URL を設定します。
license-error-page	ログインライセンスがない場合のデフォルト画面 URL を設定します。
group-error-page	ログイングループが存在しない場合のデフォルト画面 URL を設定します。
permission-error-page	無許可アクセスページにアクセスした場合のデフォルト画面 URL を設定します。
session-error-page	セッション情報不整合の場合のデフォルト画面 URL を設定します。
session-timeout-page	セッションタイムアウトの場合のデフォルト画面 URL を設定します。
system-error-page	システムエラー発生時のデフォルト画面 URL を設定します。
certification-error-page.クライアントタイプ	クライアントタイプ別に認証処理失敗時の画面 URL を設定します。
locked-error-page.クライアントタイプ	クライアントタイプ別にユーザがロック状態の場合の画面 URL を設定します。
license-error-page.クライアントタイプ	クライアントタイプ別にログインライセンスがない場合の画面 URL を設定します。
group-error-page.クライアントタイプ	クライアントタイプ別にログイングループが存在しない場合の画面 URL を設定します。
permission-error-page.クライアントタイプ	クライアントタイプ別に無許可アクセスページにアクセスした場合の画面 URL を設定します。
session-error-page.クライアントタイプ	クライアントタイプ別にセッション情報不整合の場合の画面 URL を設定します。
session-timeout-page.クライアントタイプ	クライアントタイプ別にセッションタイムアウトの場合の画面 URL を設定します。
system-error-page.クライアントタイプ	クライアントタイプ別にシステムエラー発生時の画面 URL を設定します。
request-header-certification-error-page. リクエストヘッダ値	リクエストを送信したユーザエージェント別に認証処理失敗時の画面 URL を設定します。
request-header-locked-error-page.リクエストヘッダ値	リクエストを送信したユーザエージェント別にユーザがロック状態の場合の画面 URL を設定します。
request-header-license-error-page.リクエストヘッダ値	リクエストを送信したユーザエージェント別にログインライセンスがない場合の画面 URL を設定します。

request-header-group-error-page.リクエストヘッダ値	リクエストを送信したユーザエージェント別にログイングループが存在しない場合の画面 URL を設定します。
request-header-permission-error-page.リクエストヘッダ値	リクエストを送信したユーザエージェント別に無許可アクセスページにアクセスした場合の画面 URL を設定します。
request-header-session-error-page.リクエストヘッダ値	リクエストを送信したユーザエージェント別にセッション情報不整合の場合の画面 URL を設定します。
request-header-session-timeout-page.リクエストヘッダ値	リクエストを送信したユーザエージェント別にセッションタイムアウトの場合の画面 URL を設定します。
request-header-system-error-page.リクエストヘッダ値	リクエストを送信したユーザエージェント別にシステムエラー発生時の画面 URL を設定します。

◆ ページ決定のルール

クライアントタイプ別に遷移先エラーページを設定することが可能です。

また、リクエストを送信したユーザエージェント別に遷移先エラーページを設定することも可能です。

(リクエストを送信したユーザエージェントの判定は、リクエストヘッダで行います)

これにより、リクエストの送信者(Flash、マスカット、ImAjax etc)ごとに、エラー発生時のレスポンスのコンテントタイプを変更することが出来ます。

例えば、「session-timeout-page」の場合は以下の順序でエラーページの URL が決定されます。

① **request-header-session-timeout-page.リクエストヘッダ値 の URL**

「.」 以降の文字列が、リクエストヘッダに含まれている場合

(リクエストヘッダは、小文字で判定されます)

② **session-timeout-page.クライアントタイプ の URL**

「.」 以降の文字列が、ログインユーザのクライアントタイプの場合

③ **session-timeout-page の URL**

上記以外の場合

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ **jp.co.intra_mart.foundation.security.certification.StandardSuperUserPageProvider クラス**

標準のシステム管理者ユーザページプロバイダクラス。
初期化パラメータにより、各ページの URL を返却します。

<super-security>の<page-provider>で設定するクラスです。

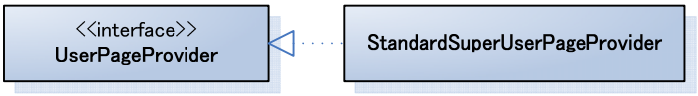


図 67 StandardSuperUserPageProvider クラス

◆ 初期化パラメータ

StandardSuperUserPageProvider クラスでは、返却するページをパラメータとして設定可能です。

表 69 StandardSuperUserPageProvider クラスのパラメータの一覧

パラメータ名	説明
initial-page	初期画面の URL を設定します。
main-page	メイン画面の URL を設定します。

■ **jp.co.intra_mart.foundation.security.certification.StandardGroupSuperUserPageProvider クラス**

標準のログイングループ管理者用ページプロバイダクラス。
初期化パラメータにより、各ページの URL を返却します。

<group-super-security>の<page-provider>で設定するクラスです。

図 68 StandardGroupSuperUserPageProvider クラス

◆ 初期化パラメータ

StandardGroupSuperUserPageProvider クラスでは、返却するページをパラメータとして設定可能です。

表 70 StandardGroupSuperUserPageProvider クラスのパラメータの一覧

パラメータ名	説明
initial-page	初期画面の URL を設定します。
main-page	メイン画面の URL を設定します。

■ **jp.co.intra_mart.foundation.security.certification.StandardUserPageProvider クラス**

一般ユーザ用ページプロバイダクラス。

初期化パラメータにより、各ページの URL を返却します。

<user-security>の<page-provider>で設定するクラスです。

図 69 StandardUserPageProvider クラス

◆ 初期化パラメータ

StandardUserPageProvider クラスでは、返却するページをパラメータとして設定可能です。

セッションの状態によって、返却するページを替えることができます。

表 71 StandardUserPageProvider クラスのパラメータの一覧

パラメータ名	説明
initial-page	デフォルトの初期画面の URL を設定します。
initial-page.クライアントタイプ	クライアント別にデフォルトの初期画面の URL を設定します。
initial-page.クライアントタイプ.ログイングループ ID	ログイングループ別にデフォルトの初期画面の URL を設定します。
main-page	デフォルトのメイン画面の URL を設定します。
main-page.クライアントタイプ	クライアント別にデフォルトのメイン画面の URL を設定します。
main-page.クライアントタイプ.ログイングループ ID	ログイングループ別にデフォルトのメイン画面の URL を設定します。

◆ ページ決定のルール

例） main-page.クライアントタイプ.[ログイングループ ID]を例に取った場合

ユーザがログイングループ ID（group_01）、クライアントタイプ（large）の場合は

① main-page.large.group_01 のURL

② main-page.large のURL

③ main-page のURL

の順序でURLを決定します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ **jp.co.intra_mart.foundation.portal.common.security.PortalStandardUserPageProvider クラス**

ポータルユーザページプロバイダクラス。

初期画面をグローバルポータルとして表示するユーザページプロバイダクラスです。
グローバルポータルが設定されている場合、初期画面をグローバルポータルに変更します。
メイン画面は常に null を返します。

<user-security>の<page-provider>で設定するクラスです。

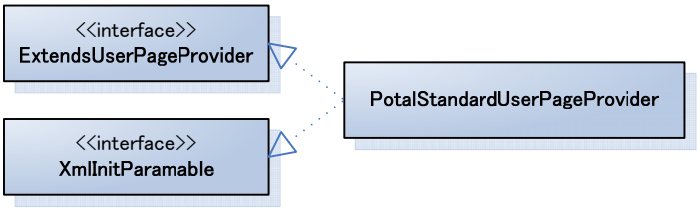


図 70 PortalStandardUserPageProvider クラス

■ **jp.co.intra_mart.foundation.security.certification.AdminUserPageProvider クラス**

グループ管理者権限ユーザ用ページプロバイダクラス
<group-super-security>の<page-provider>で設定するクラス

グループ管理者権限ユーザでログインできるログイン画面を返す。
グループ管理者権限ユーザでログインした場合に、グループ管理者権限ユーザ用のメイン画面を返す。
通常のグループ管理者でログインした場合は、メイン画面を返しません。

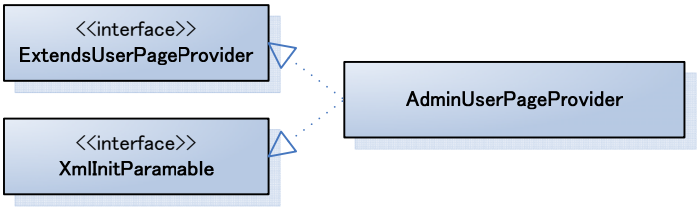


図 71 AdminUserPageProvider クラス

◆ 初期化パラメータ

AdminUserPageProvider クラスでは、返却するページをパラメータとして設定可能です。

表 72 AdminUserPageProvider クラスのパラメータの一覧

パラメータ名	説明
initial-page	初期画面の URL を設定します。
main-page	メイン画面の URL を設定します。

■ **jp.co.intra_mart.foundation.security.certification.DirectUrlUserPageProvider クラス**

ダイレクト URL ページプロバイダクラス

<user-security>の<page-provider>で設定するクラス

ダイレクト URL が有効な場合、ダイレクト URL をメイン画面として返します。
初期ページは常に null を返します。
デフォルトのクライアントタイプのみメイン画面を返します。

<user-security>の<page-provider>で設定するクラスです。

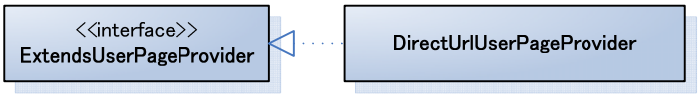


図 72 DirectUrlUserPageProvider クラス

■ **jp.co.intra_mart.foundation.security.certification.PagePatternUserPageProvider クラス**

メインページパターンページプロバイダクラス

メインページパターンが有効な場合、メインページパターンのメイン画面を返します。
初期ページは常に null を返します。
デフォルトのクライアントタイプのみメイン画面を返します。

<user-security>の<page-provider>で設定するクラスです。

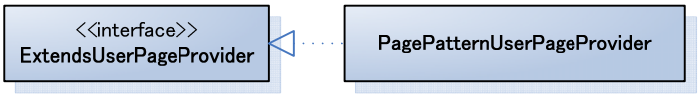


図 73 PagePatternUserPageProvider クラス

■ **jp.co.intra_mart.foundation.security.certification.ShortCutUserPageProvider クラス**

ショートカット機能ページプロバイダクラス

ショートカット ID が有効な場合、ショートカット情報に指定されているページをメインメインフレームに表示したメイン画面を返します。
初期ページは常に null を返します。
デフォルトのクライアントタイプのみメイン画面を返します。

<user-security>の<page-provider>で設定するクラスです。

図 74 ShortCutUserPageProvider クラス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

■ **jp.co.intra_mart.foundation.security.certification.PasswordHistoryUserPageProvider クラス**

パスワード履歴管理用ページプロバイダクラス

<user-security>の<page-provider>で設定するクラス

パスワード履歴管理において、パスワードの期限が切れていた場合、パスワード変更画面をメイン画面として返します。

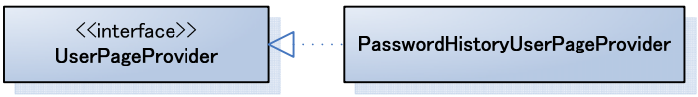


図 75 PasswordHistoryUserPageProvider クラス

■ **jp.co.intra_mart.foundation.security.certification.IntramartFrameUserPageProvider クラス**

intra-mart のフレームを利用したメインページプロバイダクラス。

intra-mart 標準のトップバーと左側にメニューを持ったメインページを返すページプロバイダです。

メインフレームエリアのページは初期化パラメータにより変更できます。

初期ページは常に null を返します。

デフォルトのクライアントタイプのみメインページを返します。

<user-security>の<page-provider>で設定するクラスです。

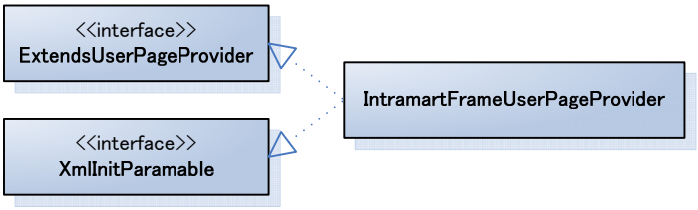


図 76 IntramartFrameUserPageProvider クラス

◆ 初期化パラメータ

表 73 IntramartFrameUserPageProvider クラスのパラメータの一覧

パラメータ名	説明
main-page	デフォルトのメイン画面の URL を設定します。 指定しない場合は system/security/user/600/main.jssp
main-frame	デフォルトのメインフレームエリアの URL を設定します。
main-frame.ログイングループ ID	ログイングループ別にデフォルトのメインフレームエリアの URL を設定します。

■ メインフレームエリアの URL 決定のルール

main-frame.[ログイングループ ID]を例に取った場合

ユーザがログイングループ ID（group_01）の場合は

① main-frame.group_01 の URL

② main-frame の URL

の順序で URL を決定します。

4.8 認証処理サーブレット

認証処理を行うサーブレット群です。

4.8.1 jp.co.intra_mart.foundation.security.servlet.SuperUserInitialServletクラス

システム管理者のログインページ要求のリクエストを受け取るサーブレットです。

デフォルトでは、/system.admin にマッピングされています。

■ SuperUserInitialServlet クラスのパラメータ

表 74 SuperUserInitialServlet クラスのパラメータの一覧

パラメータ名	説明
client-type	クライアントタイプを指定します。 指定したクライアントタイプでログイン処理を行います。 指定しなかった場合は、デフォルトのクライアントタイプを利用します。
redirect-client-types	クライアントタイプをカンマ区切りで指定します。 このパラメータに指定されたクライアントタイプで自動ログインが成功した場合は、リダイレクトを行って、メインページを表示します。 指定しなかった場合は、デフォルトのクライアントタイプを設定します。

4.8.2 jp.co.intra_mart.foundation.security.servlet.GroupSuperUserInitialServletクラス

グループ管理者のログインページ要求のリクエストを受け取るサーブレットです。

デフォルトでは、/*.manager にマッピングされています。

■ GroupSuperUserInitialServlet クラスのパラメータ

表 75 GroupSuperUserInitialServlet クラスのパラメータの一覧

パラメータ名	説明
client-type	クライアントタイプを指定します。 指定したクライアントタイプでログイン処理を行います。 指定しなかった場合は、デフォルトのクライアントタイプを利用します。
group-regexp	URL パターンからログイングループ ID を抽出する正規表現を指定します。 デフォルトは (.+)¥.
redirect-client-types	クライアントタイプをカンマ区切りで指定します。 このパラメータに指定されたクライアントタイプで自動ログインが成功した場合は、リダイレクトを行って、メインページを表示します。 指定しなかった場合は、デフォルトのクライアントタイプを設定します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.8.3 jp.co.intra_mart.foundation.security.servlet.UserInitialServlet

一般ユーザのログインページ要求のリクエストを受け取るサーブレットです。

デフォルトでは、/*.portal または、/*.mobile にマッピングされています。

■ UserInitialServlet クラスのパラメータ

表 76 UserInitialServlet クラスのパラメータの一覧

パラメータ名	説明
client-type	クライアントタイプを指定します。 指定したクライアントタイプでログイン処理を行います。 指定しなかった場合は、デフォルトのクライアントタイプを利用します。
group-regexp	URL パターンからログイングループ ID を抽出する正規表現を指定します。 デフォルトは (.+)¥.
redirect-client-types	クライアントタイプをカンマ区切りで指定します。 このパラメータに指定されたクライアントタイプで自動ログインが成功した場合は、リダイレクトを行って、メインページを表示します。 指定しなかった場合は、デフォルトのクライアントタイプを設定します。

4.8.4 jp.co.intra_mart.foundation.security.servlet.UserCertificationServlet

ログイン認証のリクエストを受け取るサーブレットです。

デフォルトでは、/user.login にマッピングされています。

■ GroupSuperUserInitialServlet クラスのパラメータ

表 77 GroupSuperUserInitialServlet クラスのパラメータの一覧

パラメータ名	説明
redirect-client-types	クライアントタイプをカンマ区切りで指定します。 このパラメータに指定されたクライアントタイプでログインが成功した場合は、リダイレクトを行って、メインページを表示します。 指定しなかった場合は、デフォルトのクライアントタイプを設定します。

4.9 ログアウトサーブレット

ログアウトを行うためのサーブレットです。

4.9.1 jp.co.intra_mart.foundation.security.servlet.LogoutServletクラス

ログアウト処理のリクエストを受け取るサーブレットです。ログアウト処理を行った後、指定の URL に遷移します。

遷移先ページの決定順位は以下の通りです。

① 1. サーブレットの引数パラメータ(im_initial_url)に指定した URL
② 2. セッションに保管されている初期 URL
③ 3. クッキーに保管されている初期 URL

この後、サーブレット初期化パラメータ(validator-class)に検証用実装クラスが
設定されている場合は、このクラスを用いて、URL の検証を行います。
なお、検証用実装クラスは、LogoutInitialUrlValidator を実装している必要があります。
URL が null の場合、および空文字列の場合は、レスポンスコード(204)を返却します。

このサーブレットでは、セッション情報を破棄し、ログイン時に表示した、ページに遷移します。
デフォルトでは、/user.logout にマッピングされています。

■ 初期化パラメータ

表 78 LogoutServlet クラスのパラメータの一覧

パラメータ名	説明
validator-class	URL 検証用実装クラスを設定します。 URL 検証用実装クラスは、指定された URL が正しいかどうか検証します。 検証用実装クラスは、LogoutInitialUrlValidator を実装している必要があります。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.10 汎用サーブレット

汎用的に利用可能なサーブレット群です。

4.10.1 jp.co.intra_mart.foundation.security.servlet.MenuServletクラス

メニューID で設定されている画面を表示するサーブレットです。

デフォルトでは、/*.menu にマッピングされています。 * には、メニューID を指定します。

ログインしているユーザが指定したメニューID の表示権限を持たない場合は、ページは表示されません。

■ 初期化パラメータ

表 79 MenuServlet クラスのパラメータの一覧

パラメータ名	説明
menu-regexp	URL パターンからメニューID を抽出する正規表現を指定します。 デフォルトは (.+)¥.
redirect-client-types	クライアントタイプをカンマ区切りで指定します。 このパラメータに指定されたクライアントタイプでログインしている場合は、リダイレクトを行って、ページを表示します。 指定しなかった場合は、デフォルトのクライアントタイプを設定します。

4.11 クライアント別URLの設定

ユーザのログインでは、クライアント（接続端末）別に URL が異なります。
新しいクライアントを追加した場合、クライアント用の URL を作成する必要があります。
クライアント用の URL の作成は、Application Runtime/doc/imart/WEB-INF/web.xml（以下 web.xml と称す）の編集が必要となります。

標準（ブラウザ）の設定内容は以下のようになっています。

```
・・・・・
<servlet>
  <servlet-name>UserInitialServlet</servlet-name>
  <servlet-class>jp.co.intra_mart.foundation.security.servlet.UserInitialServlet</servlet-class>
  <init-param>
    <param-name>client-type</param-name>
    <param-value>pc</param-value>
  </init-param>
</servlet>
・・・・・
  <filter-mapping>
    <filter-name>NoCacheFilter</filter-name>
    <servlet-name>UserInitialServlet</servlet-name>
  </filter-mapping>
  <filter-mapping>
    <filter-name>HttpSessionMonitoringFilter</filter-name>
    <servlet-name>UserInitialServlet</servlet-name>
  </filter-mapping>
  <filter-mapping>
    <filter-name>ResponseCharacterEncodingFilter</filter-name>
    <servlet-name>UserInitialServlet</servlet-name>
  </filter-mapping>
  <filter-mapping>
    <filter-name>HTTPContextHandlingFilter</filter-name>
    <servlet-name>UserInitialServlet</servlet-name>
  </filter-mapping>
  <filter-mapping>
    <filter-name>JSSPContextFilter</filter-name>
    <servlet-name>UserInitialServlet</servlet-name>
  </filter-mapping>
・・・・・
  <servlet-mapping>
    <servlet-name>UserInitialServlet</servlet-name>
    <url-pattern>*.portal</url-pattern>
  </servlet-mapping>
・・・・
・
```

- **servlet タグ**
  servlet-name はサーブレットを識別するユニークな名称です。
  servlet-class は jp.co.intra_mart.foundation.security.servlet.UserInitialServlet 固定です。
  初期 URL を受け付ける標準のサーブレットクラスです。
  init-param には‘client-type’のキー名で対象のクライアントタイプを設定します。
  ここでは、pc がブラウザ用のクライアントタイプであるので、’pc’が設定されています。
- **filter-mapping タグ**
  サーブレットに対してすべての設定が必要です。（順番も同じにする必要があります。）
  servlet-name タグには対象の servlet-name を設定します。
- **servlet-mapping タグ**
  URL パターンを設定します。ブラウザ（’pc’）の URL パターンは*.portal です。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.11.1 URLの追加

新しいクライアントタイプを作成した場合の web.xml の設定サンプル（追加部分）を提示します。
各タグの記述場所（前後関係）が決まっていますので注意してください。

クライアントタイプの設定方法は、「2.2 対応クライアント」を参照してください。

クライアントタイプ	pda
サーブレット名	UserInitialServletForPda
URL パターン	*.pda

```
<servlet>
  <servlet-name>UserInitialServletForPda</servlet-name>
  <servlet-class>jp.co.intra_mart.foundation.security.servlet.UserInitialServlet</servlet-class>
  <init-param>
    <param-name>client-type</param-name>
    <param-value>pda</param-value>
  </init-param>
</servlet>
  ・・・・・
<filter-mapping>
  <filter-name>NoCacheFilter</filter-name>
  <servlet-name>UserInitialServletForPda </servlet-name>
</filter-mapping>
<filter-mapping>
  <filter-name>HttpSessionMonitoringFilter</filter-name>
  <servlet-name>UserInitialServletForPda </servlet-name>
</filter-mapping>
<filter-mapping>
  <filter-name>ResponseCharacterEncodingFilter</filter-name>
  <servlet-name>UserInitialServletForPda </servlet-name>
</filter-mapping>
<filter-mapping>
  <filter-name>HTTPContextHandlingFilter</filter-name>
  <servlet-name>UserInitialServletForPda </servlet-name>
</filter-mapping>
<filter-mapping>
  <filter-name>JSSPContextFilter</filter-name>
  <servlet-name>UserInitialServletForPda </servlet-name>
</filter-mapping>
  ・・・・・
<servlet-mapping>
  <servlet-name>UserInitialServletForPda</servlet-name>
  <url-pattern>*.pda</url-pattern>
</servlet-mapping>
```

http://サーバ名:ポート/imart/ログイングループ ID.pda
でアクセスできます。

4.11.2 ページ作成

新しいクライアントを作成した場合は、クライアント用の以下のページを作成し、設定する必要があります。

4.11.2.1 エラーページの作成

作成したクライアント用のページを作成して、設定を追加します。

作成したページの設定は、サーバマネージャ/conf/access-security.xml で行います。

以下は、クライアントタイプ('pda')の設定追加例です。

```
<security-config>
  <error-page-provider>

<page-provider-class>jp.co.intra_mart.foundation.security.certification.StandardErrorPageProvider</page-provider-class>
        ........
        <init-param>
            <param-name>certification-error-page.pda</param-name>
            <param-value>認証エラーページパス</param-value>
        </init-param>
        <init-param>
            <param-name>locked-error-page.pda</param-name>
            <param-value>アカウントロックエラーページパス</param-value>
        </init-param>
        <init-param>
            <param-name>license-error-page.pda</param-name>
            <param-value>ライセンスエラーページパス</param-value>
        </init-param>
        <init-param>
            <param-name>group-error-page.pda</param-name>
            <param-value>ログイングループなしのエラーページパス</param-value>
        </init-param>
        <init-param>
            <param-name>permission-error-page.pda</param-name>
            <param-value>表示権限なしのエラーページパス</param-value>
        </init-param>
        <init-param>
            <param-name>session-error-page.pda</param-name>
            <param-value>セッション整合性エラーページパス</param-value>
        </init-param>
        <init-param>
            <param-name>session-timeout-page.pda</param-name>
            <param-value>セッションタイムアウトページパス</param-value>
        </init-param>
        <init-param>
            <param-name>system-error-page.pda</param-name>
            <param-value>システムエラーページパス</param-value>
        </init-param>
  </error-page-provider>
</security-config>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.11.2.2 ログインページとメインページの作成

ログインページ（初期アクセス時のページ）とメインページ（ログイン後のページ）を作成し、設定する必要があります。

作成したページの設定は、サーバマネージャ/conf/access-security.xml で行います。

以下は、クライアントタイプ(‘pda’)の設定追加例です。

```
<user-security>
……
  <page-provider>
    <page-provider-class>jp.co.intra_mart.foundation.portal.utility.StandardUserPageProvider</page-provider-class>
    ……
    <init-param>
      <param-name>initial-page.pda</param-name>
      <param-value>ログインページのパス</param-value>
    </init-param>
    <init-param>
      <param-name>main-page.pda</param-name>
      <param-value>メインページのパス</param-value>
    </init-param>
    ……
  </page-provider>
  ……
</user-security>
```

4.11.2.3 メニュー用ページの作成

新しく作成したクライアント用の各種ページを作成し、メニュー管理の設定画面から登録します。

4.12 ログイン画面の設定

ログイン画面に関する設定です。

Server Manager/conf/ parameter.xml（以下 parameter.xml と称す）の編集が必要となります。

4.12.1 ユーザ名の 表示設定

ログイン画面の表示の際に、前回ログインしたユーザ名を[表示する]／[表示しない]が設定できます。

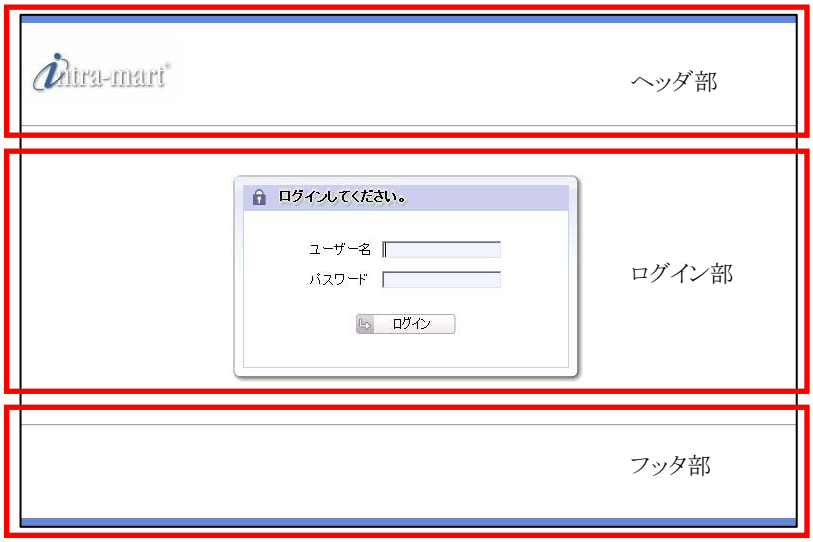
```
<param>
    <param-name>im.user.valid</param-name>
    <param-value>ON</param-value>
</param>
```

<param-value>タグに、「ON」および「OFF」のどちらかを設定します。

ON	前回ログインしたユーザ名が表示されます。
OFF	ユーザ名は空白で表示されます。

4.12.2 ログイン画面ヘッダ・フッタの指定

ログイン画面は、以下のような 3 部構成となっています。



ログイン画面のヘッダおよびフッタは、作成した任意の画面を設定することができます。

```
<param>
    <param-name>im.standard.login.header</param-name>
    <param-value>system/security/common/standard_login_header</param-value>
</param>
```

<param-value>タグには、[Resource Service]からのパス（拡張子なし）を設定します。

ログイン画面のヘッダおよびフッタを表示させない場合は、<patam>タグをコメントアウトします。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

4.13 タイトルバーの設定

タイトルバーに関する設定です。
Server Manager/conf/ parameter.xml（以下 parameter.xml と称す）の編集が必要となります。

タイトルバーに表示されるロゴ（イメージ）を設定できます。
設定を行っていない場合は、intra-mart のロゴ（イメージ）が表示されます。

4.13.1 システム管理者

システム管理者でログインを行った場合に表示されるロゴ（イメージ）を設定します。
カラーパターンに対応しています。
カラーパターンに対応したロゴ（イメージ）を用意することにより、カラーパターンに対応したロゴ（イメージ）を表示することもできます。

[WebServerConnector]また、[Application Runtime（intra-mart HttpServer を使用する場合）]からのパスを設定します。
カラーパターンに対応したロゴ（イメージ）を配置する場合は、[%COLORPATTERN%]を設定することにより「カラーパターン ID」に置換されます。

```
<param>
    <param-name>system.header.logo5</param-name>
    <param-value>skin/%COLORPATTERN%/logo2.gif</param-value>
</param>
```

ロゴ（イメージ）を表示なしの場合は、<param-value> タグに[%NOIMAGE%]と設定してください。

例）ロゴ（イメージ）を表示なしの場合

```
<param>
    <param-name>system.header.logo5</param-name>
    <param-value>%NOIMAGE%</param-value>
</param>
```

4.13.2 グループ管理者／ 一般ユーザ（メインページパターン：MainPage510）

グループ管理者および一般ユーザ（メインページパターン：MainPage510）でログインを行った場合に表示されるロゴ（イメージ）を設定します。
カラーパターンに対応しています。
カラーパターンに対応したロゴ（イメージ）を用意することにより、カラーパターンに対応したロゴ（イメージ）を表示することもできます。

ログイングループごとに、ロゴ（イメージ）を設定します。
ログイングループに対応した設定を行っていない場合は、intra-mart のロゴ（イメージ）が表示されます。

```
<param>
    <param-name><%ログイングループ ID%>.group.header.logo5</param-name>
    <param-value>skin/%COLORPATTERN%/logo2.gif</param-value>
</param>
```

<param-name>タグの<%ログイングループ ID%>には、ログイングループ ID を指定します。
<param-value>タグには、表示されるロゴ（イメージ）を設定します。

```
例）ログイングループ ID が「default」の場合の設定

<param>
    <param-name>default.group.header.logo5</param-name>
    <param-value>skin/%COLORPATTERN%/logo2.gif</param-value>
</param>
```

ロゴ（イメージ）を表示なしの場合は、<param-value> タグに[%NOIMAGE%]と設定してください。

4.13.3 一般ユーザ（メインページパターン：システムデフォルト）

一般ユーザ（メインページパターン：システムデフォルト）でログインを行った場合に表示されるロゴ（イメージ）を設定します。
カラーパターンに対応しています。
カラーパターンに対応したロゴ（イメージ）を用意することにより、カラーパターンに対応したロゴ（イメージ）を表示することもできます。

ログイングループごとに、ロゴ（イメージ）を設定します。
ログイングループに対応した設定を行っていない場合は、intra-mart のロゴ（イメージ）が表示されます。

```
<param>
    <param-name><%ログイングループ ID%>.group.header.logo6</param-name>
    <param-value>skin/%COLORPATTERN%/logo1.gif</param-value>
</param>
```

<param-name>タグの<%ログイングループ ID%>には、ログイングループ ID を指定します。
<param-value>タグには、表示されるロゴ（イメージ）を設定します。

ロゴ（イメージ）を表示なしの場合は、<param-value> タグに[%NOIMAGE%]と設定してください。

```
例）ログイングループ ID が「default」の場合の設定

<param>
    <param-name> default.group.header.logo6</param-name>
    <param-value>skin/%COLORPATTERN%/logo1.gif</param-value>
</param>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5 拡張機能

5.1 ダイレクトURL

ダイレクト URL を使用することで、ログイン後のメインページを自由に切り替えることができます。
ダイレクト URL の指定は以下のように、URL パラメータで指定します。
(注)ダイレクト URL はモバイルでのアクセスには対応しておりません。

http://ホスト名/imart/default.portal?im_url=ダイレクト URL&im_url_sign=URL 署名

ダイレクト URL	指定したい URL を URL エンコードしたもの
URL 署名	AccessSecurity.createUrlSignature メソッドを使用して作成。

5.1.1 ダイレクトURLパラメータの作成

例えば、pages/src/sample/direct.js および direct.html を指定する場合の作成手順を説明します。

ダイレクト URL	sample%2fdirect.jssp (sample/direct.jssp を URL エンコードしたもの)
URL 署名	AccessSecurity.createUrlSignature(String url, String groupId, String userId); url : sample/direct.jssp（注：URL エンコードしたものではないです。） groupId : 対象のログイングループ userId : 対象のユーザ ID

上記のパラメータを指定して、ログイン画面を呼び出し、正常にログインできた場合、ログイン後に指定したダイレクト URL のページが表示されます。

指定したダイレクト URL 表示されるためには、以下の条件が必要です。

- im_url の値（URL デコード後）が署名を作成した時の url と同じであること。
- ログインするログイングループが署名を作成した時の groupId と同じ。
- ログインするユーザが署名を作成した時の userId と同じ。

1 つでも一致しない場合は、通常のログイン後の画面が表示されます。
URL の詐称は出来ない仕組みになっています。

5.1.2 ダイレクトURLを使用する場合の注意点

指定した URL が表示される場合は、そのページ自体が表示されるので、トップバーやメニュー画面はありません。
表示した画面のリンクなどで、メインページへ誘導できるようにしていただく必要があります。

5.2 ショートカットアクセス機能

ショートカットアクセス機能は、初期アクセス URL にショートカットアクセス用の URL パラメータを指定することによって、ログイン後の画面を任意の画面に取り替えることができる機能です。
ショートカットアクセス機能はダイレクト URL の上位機能として提供されています。
ショートカットアクセス機能を用いると、ダイレクト URL がトップバーおよびメニュー画面が存在しないのに対し、トップバーおよびメニュー画面が存在し、そのメインエリアに任意のページを表示することが可能になります。
また、表示に関するセキュリティも設定することが可能です。
ショートカットアクセス機能を用いた場合に利用できるセキュリティ機能は以下の通りです。

- 指定ページを表示できる許可ユーザの指定。（複数設定可能）
- 指定ページを表示できる有効期間の指定。
- ログインの制御に関して以下の機能を選択できます。
- ログイン画面からユーザ ID、パスワードを入力後、直接指定ページにアクセス。
- ユーザ ID、パスワードを指定せずに、直接ページにアクセス可能。
  （表示許可ユーザを一名のみ指定した場合のみ利用できる機能です。）
- 許可ユーザの指定と合わせて、ユーザの許可をプログラムにより行う拡張検証機能

5.2.1 ショートカットアクセスURL

ショートカットアクセス用の URL は、通常の初期アクセス URL にショートカット ID をパラメータとして追加した URL です。

http://<server>/<context-path>/<login-group>.portal?im_shortcut=<ショートカット ID>

例：
http://localhost/imart/default.portal?im_shortcut=xazh03nbe43wd

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

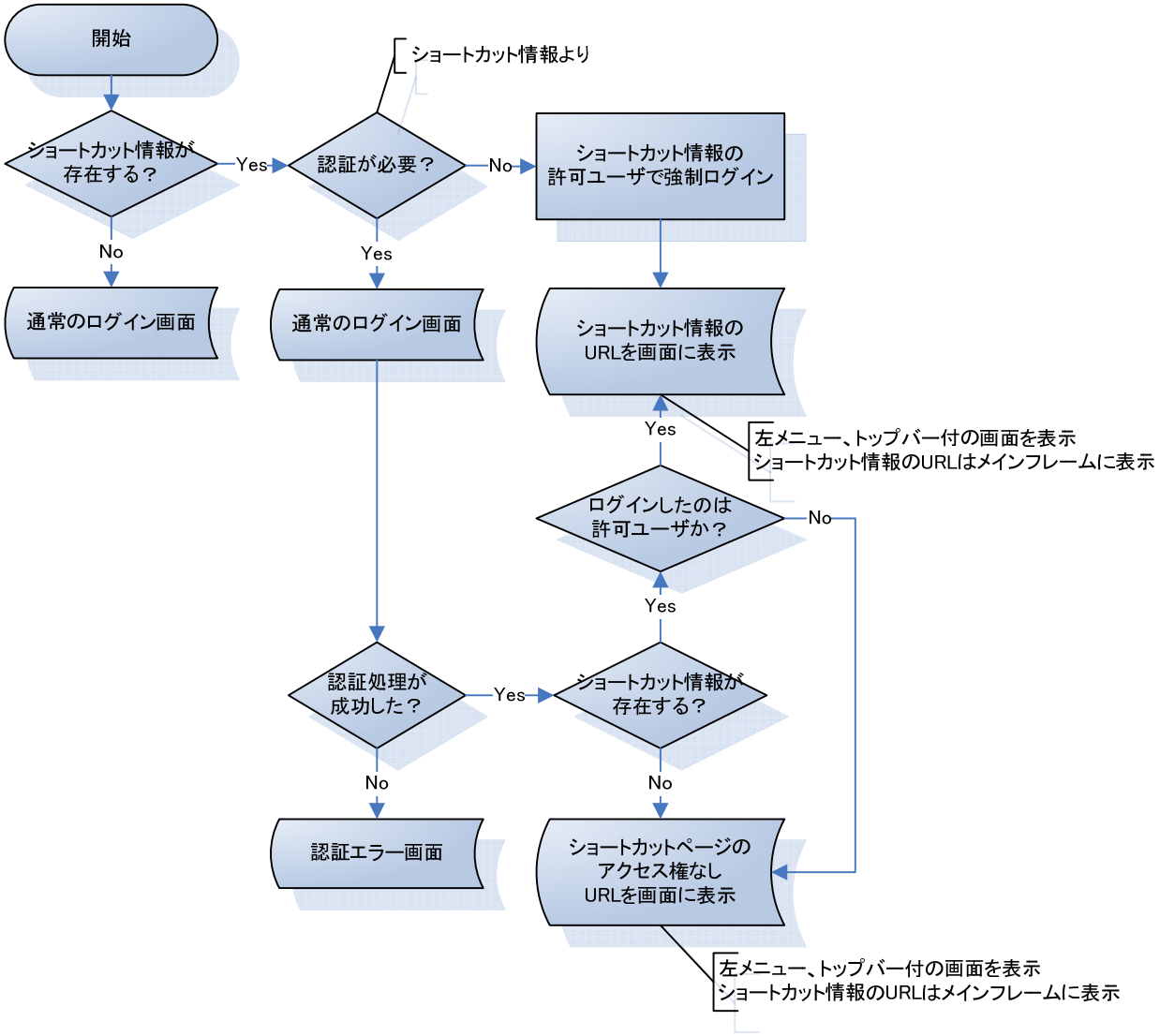
5.2.2 ショートカットIDの作成

ショートカット ID は、表示するページの情報およびセキュリティの情報に紐づく ID となります。
ショートカット ID は、表示するページの情報およびセキュリティの情報を指定して API を用いて作成します
メインエリアに、pages/src/sample/shortcut.js および shortcut.html を指定する場合のショートカット ID の作成手順を説明します。

```
// ショートカットマネージャの作成
var manager = new ShortCutManager("default");

// ショートカット情報の作成
var shortCutInfo = new Object();

  // 表示する URL
  shortCutInfo.url = "sample/shortcut.jssp";

  // 表示する URL に渡すパラメータの設定(任意指定)
  shortCutInfo.urlParams = new Object();
  shortCutInfo.urlParams["arg1"] = "value1";
  shortCutInfo.urlParams["arg2"] = "value2";

  // 表示許可を行うユーザ
  shortCutInfo.allowUsers = new Array("guest","ueda");

  // ログイン認証が必要かどうか？(認証必要)
  shortCutInfo.isAuth = true;

  // この情報の有効期限(作成時から10日間有効)
  shortCutInfo.validEndDate = manager.addValidEndDate(10);


// ショートカットID 作成
var shortCutId = manager.createShortCut(shortCutInfo);
```

説明はスクリプト開発モデルですが、JavaEE 開発モデルにも同様の API が存在します。

5.2.3 ショートカットアクセス機能を利用した場合の画面遷移



5.2.4 ショートカット拡張検証機能

拡張検証機能では、ショートカット情報の URL の表示を許可ユーザ以外で、ログインしたユーザに対して検証プログラム利用して、ページの表示許可を与えるかどうかの判定を追加で行う機能です。

拡張検証コードは、検証プログラムのコード名を表します。

拡張検証パラメータは、検証プログラムに渡されるパラメータとなります。

検証プログラムとコード名の紐付けは、ショートカット情報操作設定の定義ファイルで行います。

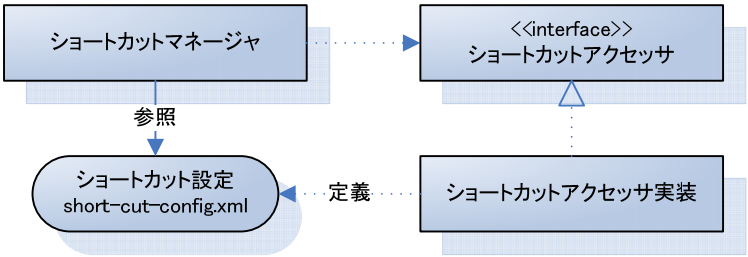
Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.2.5 ショートカットマネージャ

ショートカットマネージャでははは、ショートカット情報の管理を行うマネージャクラスです。
スクリプト開発モデルでは ShortCutManager
JAVAEE 開発モデルでは jp.co.intra_mart.foundation.security.shortcut.ShortCutManager
です。

5.2.6 ショートカット情報操作アーキテクチャ

ショートカットマネージャはショートカット設定に従い、ショートカット操作モジュール（ショートカットアクセッサ）から
ショートカット情報を操作します。



5.2.7 ショートカット情報操作設定の定義

ショートカット情報操作モジュールの設定を行います。
設定ファイルは、ServerManager/conf/short-cut-config.xml に記述します。
設定例を以下に示します。
この設定を元にサーバマネージャはショートカット情報操作モジュールの実装を取得します。

```
<short-cut-config>

  <short-cut-accessor>
    <short-cut-accessor-class>SampleShortcutAccessor</short-cut-accessor-class>
  <error-page>sample.jssp</error-page>
    ....
  </short-cut-accessor>

</short-cut-config>
```

■ タグの説明

<short-cut-accessor>		ショートカット操作モジュールに関する設定を行います。
	<short-cut-accessor-class>	ショートカット情報を操作するモジュール（ショートカットアクセッサ）の実装クラスを設定します。 jp.co.intra_mart.foundation.security.shortcut.ShortCutAccessor インタフェースを実装する必要があります。
	<validator>	検証コードと検証プログラムの紐付けを行います。（複数設定可能） 属性は以下の通りです。 code： 検証コードを指定します。 class： 検証プログラムのクラスを指定します。 jp.co.intra_mart.foundation.security.shortcut.ShortCutValidator インタフェースを実装する必要があります。
	その他任意のタグ	この実装クラスの初期化時に<short-cut-accessor>をルートとした DOM を引数として渡します。 このタグ内に任意のタグを指定できます。

5.2.8 ショートカット情報

ショートカット情報は、ショートカット ID と紐づくさまざまな情報を格納するオブジェクトです。

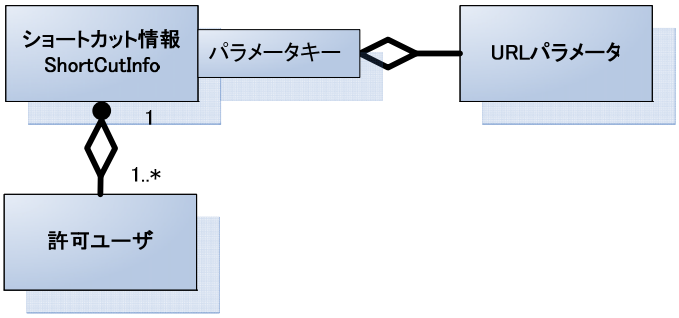
項目	説明	型	プロパティ
URL	ショートカットとして表示する URL	String	url
URL 引数	URL に渡す引数情報	Java ： Map JavaScript ： Object	urlParams
許可ユーザ	ショートカットページの表示を 許可するユーザ	Java ： String[] JavaScript ：Array(String)	allowUsers
認証フラグ	認証が必要かどうかのフラグ true ： 認証必要 / false ： 認証不要 許可ユーザが 1 名の時だけ、 false にできる。	Boolean	isAuth
拡張検証コード	追加でユーザが許可されるかどうかの 検証コード	Strimg	validationCode
拡張検証パラメータ	検証処理に渡される追加のパラメータ	String	validationParam
有効終了日	有効終了日	Date	validEndDate

5.2.9 ショートカット情報のモデル

- **jp.co.intra_mart.foundation.security.shortcut.ShortCutInfo クラス**

  ショートカット情報を格納するクラスです。

  ショートカット情報については、このクラスを通して、アクセスします。



5.2.10 ショートカット情報操作モジュールインタフェース

- **jp.co.intra_mart.foundation.security.shortcut.ShortCutAccessor インタフェース**

  ショートカット情報を操作するリモジュールはこのインタフェースを実装する必要があります。

  ショートカット情報を操作するインタフェースを持ちます。

- **jp.co.intra_mart.foundation.security.shortcut.ShortCutValidator インタフェース**

  ショートカットのページの表示を許可すかどうかを検証する検証プログラムには、このインタフェースを実装する必要があります。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.2.11 ショートカット情報操作実装モジュール

■ **jp.co.intra_mart.foundation.security.shortcut.StandardShortCutAccessor クラス**

データベース経由で、ショートカット情報を操作する標準実装のモジュールです。

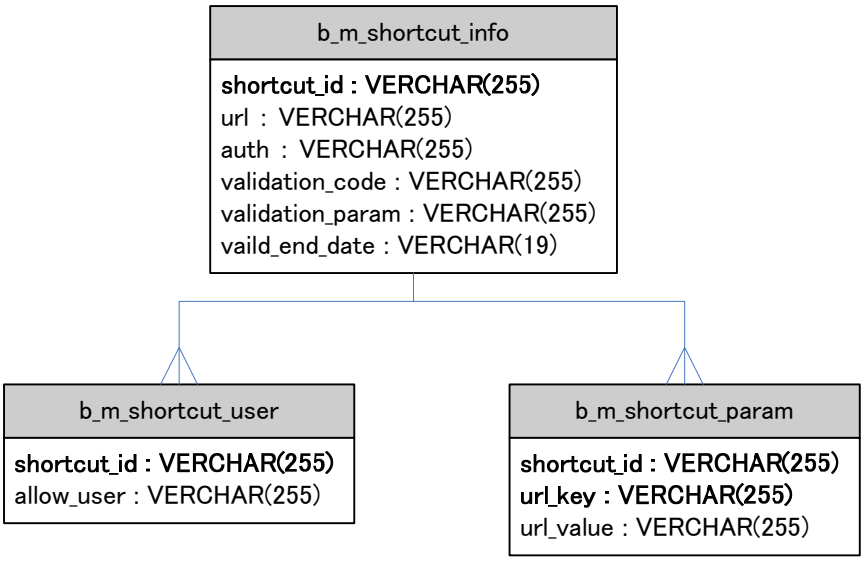
図 77 StandardShortCutAccessor クラス

◆ 初期化パラメータ

タグ名	説明
error-page	ショートカット情報が存在しない場合に表示するページパスを指定します。
main-page	ショートカットページを表示するためのメインページパスを設定します。

◆ ショートカット情報 ER 図

ショートカット ID とショートカット情報を紐付けて、DB に保管します。



■ **jp.co.intra_mart.foundation.security.shortcut.ShortCutValidator クラス**

拡張検証パラメータに指定された正規表現にマッチするユーザ ID のユーザを許可ユーザとする実装クラスです。

標準で定義されている検証プログラムです。

検証コード ： **RegExpUser**

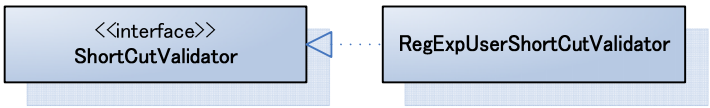


図 78 RegExpUserShortCutValidator クラス

■ **jp.co.intra_mart.foundation.security.shortcut.RoleUserShortCutValidator クラス**

拡張検証パラメータに指定されたロール ID を持つユーザを許可ユーザとする実装クラスです。

標準で定義されている検証プログラムです。

検証コード ： **RoleUser**

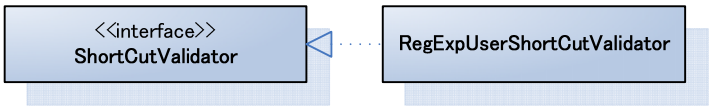


図 79 RoleUserShortCutValidator クラス

■ **jp.co.intra_mart.foundation.security.shortcut.ScriptShortCutValidator クラス**

スクリプトを実行して許可ユーザを検証する実装クラスです。

ショートカット情報の拡張検証パラメータに指定したスクリプトの isAllowUser メソッドで判定します。

このモジュールを利用する場合は、ショートカット情報の拡張検証パラメータにスクリプトのパスを指定します。拡張子はつけません。

例 ： shortcut/validator

また、スクリプトの isAllowUser メソッドに対して、パラメータ(param)を渡したい場合は、ショートカット情報の拡張検証パラメータのスクリプトのパスに続いて、カンマ区切りで、パラメータを指定します。

例 ： shortcut/validator,パラメータ値

パラメータ値が指定されていない場合は、スクリプトの isAllowUser メソッドの param 引数には空文字列が渡されます。

スクリプトの isAllowUser メソッドのインタフェース

Boolean isAllowUser(String groupId, ShortCutInfo shortcutInfo,String userId,String param)

標準で定義されている検証プログラムです。

検証コード ： **Script**

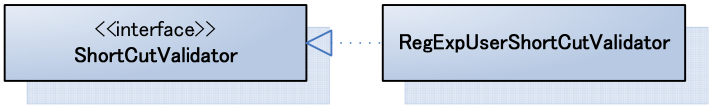


図 80 ScriptShortCutValidator クラス

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.3 パスワード履歴管理機能

パスワード履歴管理では、パスワードの履歴管理を行い、パスワードの定期的な変更を促す機能を提供します。
また、パスワードの様々な制約を設定することが可能です。
標準実装では以下の機能が利用できます。

- パスワード有効期間の設定
- 初回ログイン時のパスワード変更要求
- ユーザ ID と同じパスワードの禁止
- パスワードの長さの指定
- パスワードの使用許可文字の制御
- 同じパスワードの繰り返し利用の禁止（パスワード履歴）
- ポータル（汎用新着ポートレット）への有効期限切れ警告表示。

5.3.1 パスワード履歴管理機能における注意事項

パスワード履歴管理では、システムに保管されているユーザのパスワードについての管理を行います。
LDAP による認証を行う場合は、LDAP に対してパスワードの変更処理が行えないため、この機能はご利用できません。

5.3.2 パスワード履歴管理アーキテクチャ

パスワード履歴管理を行うモジュール（実装モジュール）は取替え可能なモジュールです。

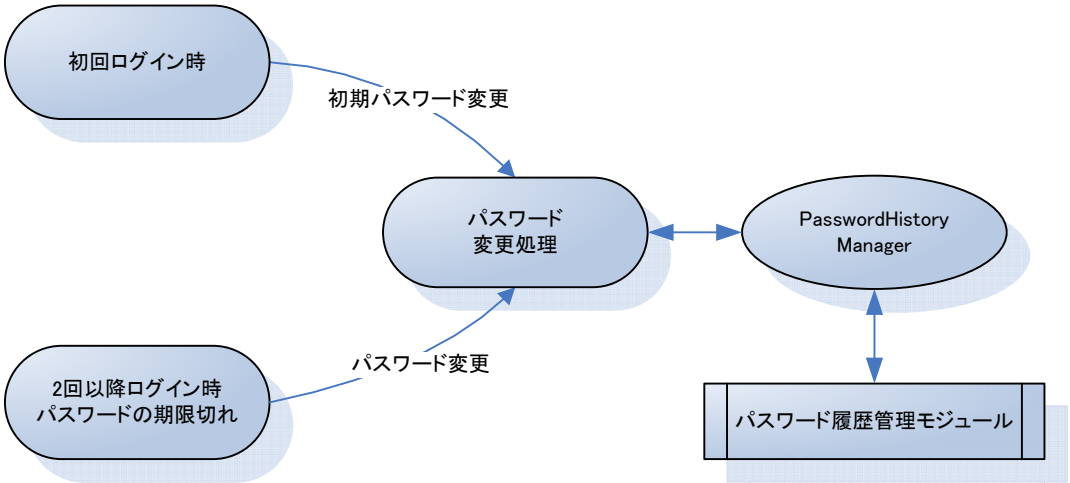


図 81 パスワード履歴管理アーキテクチャ

5.3.3 パスワード履歴管理マネージャ

パスワード履歴管理マネージャは、パスワードのチェックおよび、パスワード履歴管理を行うマネージャクラスです。

スクリプト開発モデルでは PasswordHistoryManager

JavaEE 開発モデルでは jp.co.intra_mart.foundation.security.password. PasswordHistoryManager

です。

5.3.4 パスワード履歴管理情報設定の定義

パスワード履歴管理情報設定では、パスワード履歴管理モジュールの設定およびそのモジュールで使用する各種設定を行います。

設定ファイルは、ServerManager/conf/password-history.xml に記述します。

設定例を以下に示します。

```
<password-history>
  <group-default accessor-class="StandardPasswordHistoryAccessor">
        ･････････
  </group-default>
  <group name="sample" accessor-class="StandardPasswordHistoryAccessor">
        ･････････
  </group>
</password-history>
```

■ タグの説明

<group-default>		各ログイングループのデフォルト設定を行います。(必須項目) group タグの name 属性に設定されたログイングループはその group タグ内の設定が優先されます。
	<accessor-class>	パスワード履歴管理モジュールの実装クラスを設定します。 jp.co.intra_mart.foundation.security.password.PasswordHistoryAccessor インタフェースを実装している必要があります。 省略した場合は、 jp.co.intra_mart.foundation.security.password.StandardPasswordHistoryAccessor を設定します。

<group>		ログイングループ単位の設定を行います。(任意項目) group タグの name 属性に設定されたログイングループはその group タグ内の設定が優先されます。
	<name>	対象となるログイングループ ID を設定します。(必須項目)
	<accessor-class>	パスワード履歴管理モジュールの実装クラスを設定します。 jp.co.intra_mart.foundation.security.password.PasswordHistoryAccessor インタフェースを実装している必要があります。 省略した場合は、group-default タグの accessor-class 属性に設定したクラスが使用されます。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.3.5 パスワード履歴情報

パスワード履歴として保管される情報です。

表 80 パスワード履歴情報

項目	説明
パスワード	パスワード
登録日付	パスワード登録日付

■ パスワード履歴管理処理結果情報

パスワード履歴管理に関する処理を行った結果の情報です。

表 81 パスワード履歴管理処理結果情報

項目	説明
処理フラグ	処理が成功した場合は true
エラーメッセージ	エラーが発生した場合のエラーメッセージ
詳細エラーメッセージ	エラーが発生した場合の詳細なメッセージ

5.3.6 パスワード履歴管理のモデル

■ **jp.co.intra_mart.foundation.security.password.PasswordHistory** クラス

パスワード履歴情報を提供するクラスです。

パスワード履歴は必ずこのクラスを通して操作します。

■ **jp.co.intra_mart.foundation.security.password.PasswordHistoryResult** クラス

パスワード履歴管理の処理結果の情報を提供するクラスです。

5.3.7 パスワード履歴管理操作モジュールインタフェース

■ **jp.co.intra_mart.foundation.security.password.PasswordHistoryAccessor** インタフェース

パスワード履歴情報の管理およびパスワードのチェックを行うためのインタフェースです。

5.3.8 パスワード履歴管理操作モジュール実装クラス

■ **jp.co.intra_mart.foundation.security.password.StandardPasswordHistoryAccessor クラス**

標準のパスワード履歴情報の管理およびパスワードのチェックを行うための実装クラスです。
このクラスでは下記の図の処理を行います。

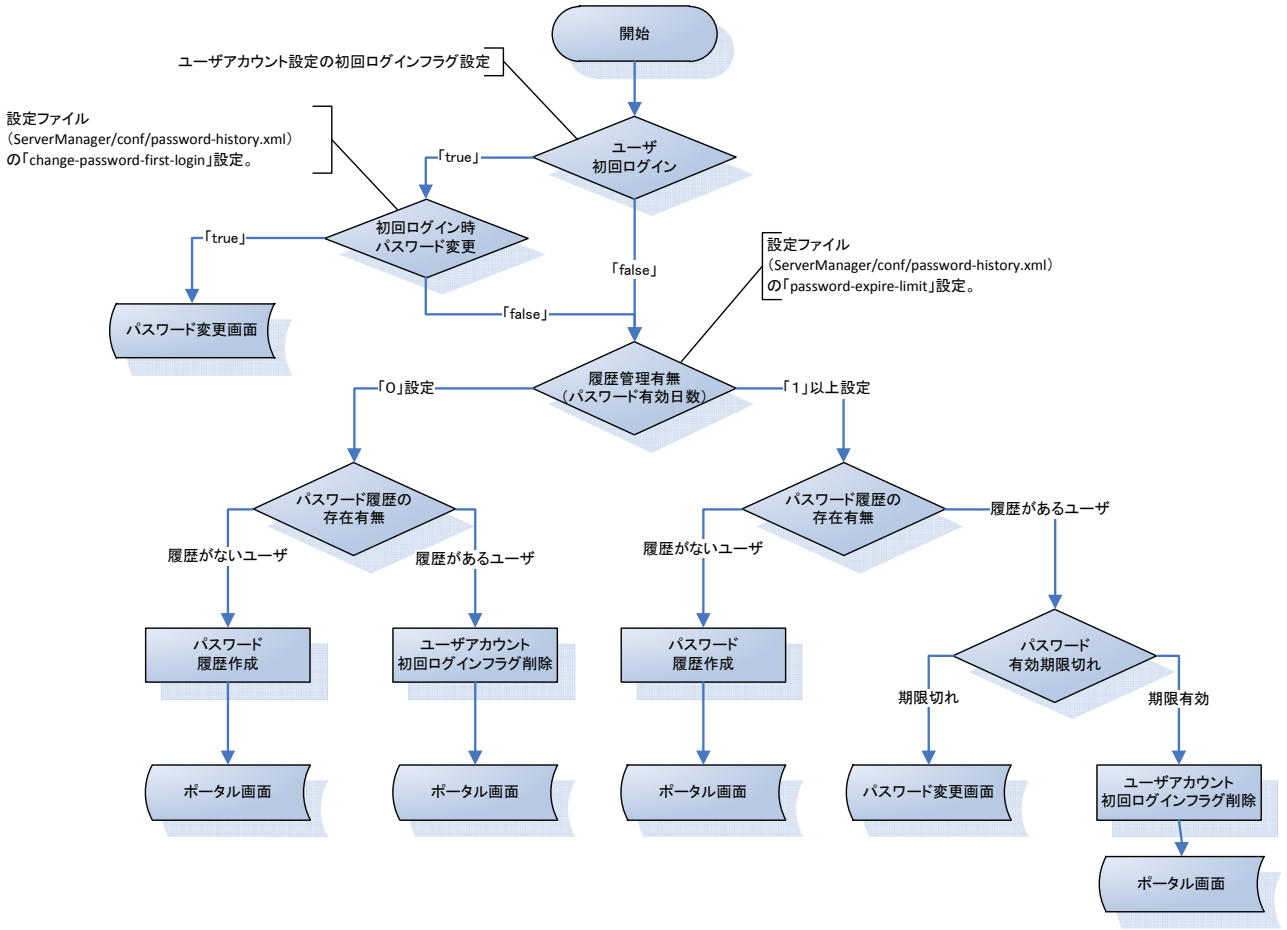


図 82 StandardPasswordHistoryAccessor クラスの処理フロー図

このクラス名は設定ファイル(ServerManager/conf/password-history.xml)に記述します。
設定するタグは、<group-default>の accessor-class 属性かまたは、<group>の accessor-class 属性に記述します。

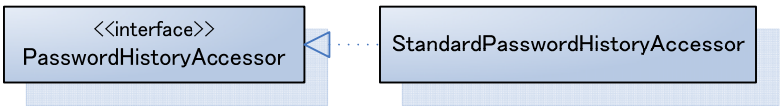


図 83 StandardPasswordHistoryAccessor クラス

◆ 初期化パラメータ

パラメータ実装例は以下の通りです。
<group-default>タグと<group>タグには同じ設定項目が存在します。
ただし、ログインしているログイングループと<group>タグの name 属性値が同じタグ内の設定が優先して使用されます。
一致する<group>タグがないまたは、タグ内の項目が存在しない場合は、<group-default>タグの設定

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

が利用されます。

```
<password-history>
  <group-default accessor-class="SampleStandardPasswordHistoryAccessor">
    <change-password-first-login>true</change-password-first-login>
    <password-expire-limit>30</password-expire-limit>
    <password-history-count>5</password-history-count>
    <deny-client-types>mobile</deny-client-types>
    <password-expire-page>/system/security/user/password_expire.jssp</password-expire-page>
    <check-password enable="false">
      <check-password-length enable="false" min="0" max="50"/>
      <allow-latin-letters required="false">ABCDEFGHI……</allow-latin-letters>
      <allow-number required="false">0123456789</allow-number>
      <allow-extra-char required="false">_-.+$#!/@</allow-extra-char>
      <deny-old-password>false</deny-old-password>
      <deny-userid>false</deny-userid>
    </check-password>
    <password-cryption-class>SampleStandardCryption</password-cryption-class>
    <return-initial-page>false</return-initial-page>
  </group-default>
  <group name="default">
      ……
  </group>
</password-history>
```

表 82 StandardPasswordHistoryAccessor クラスのパラメーター覧

タグ名	説明	デフォルト
<change-password-first-login>	初期ログイン時にパスワード変更を要求するかどうかを指定します。 ユーザ属性の初回ログイン設定の値をチェックして、初回ログイン判定を行います。	true
<password-expire-limit>	パスワードの有効日数。 0を指定した場合、有効期間は無制限です。	0
<password-history-count>	パスワードの履歴を管理する世代。 0を指定した場合、履歴管理を行いません。	0
<deny-client-types>	パスワード履歴管理対象とならないクライアントタイプの一覧。 複数指定する場合は、カンマ区切りで指定します。	mobile
<password-expire-page>	パスワード変更画面のパスを指定します。 スクリプト開発モデルのパスの場合は、拡張子に.jssp を付加します。	*1
<check-password>	パスワード文字列に対する有効チェックの内容を設定します。 設定詳細は「表 83 check-passwordタグ内のパラメーター覧」を参照。	なし
	enable 属性 パスワード文字列の有効チェックを行うかどうかを設定します	false
<password-cryption-class>	パスワードの履歴を保管する際に用いられる暗号化アルゴリズムクラス。 設定するクラスは、 jp.co.intra_mart.foundation.security.cryption.Cryption インタフェースを実装する必要があります。	*2
<return-initial-page>	パスワード変更要求画面よりパスワードを変更した後に、ログイン画面に戻るかどうかを指定します。	false

*1　デフォルトは/system/security/user/password_expire.jssp です。この画面ではユーザアカウント設定の初回ログインフラグが有効の場合、初回ログイン画面として表示されます。無効の場合、パスワード有効期限切れ画面として表示されます。

*2　デフォルトは jp.co.intra_mart.foundation.security.cryption.StandardCryption です。

表 83 check-password タグ内のパラメーター覧

タグ名	説明	デフォルト
<check-password-length>	パスワードの長さ制約に関する設定です。	なし
	enable 属性　パスワードの長さ制約を行うかどうかを設定します。	false
	min 属性　パスワードの最小の長さを設定します。最小値は 0。*1	0
	max 属性　パスワードの最大の長さを設定します。最大値は 50。	50
<allow-latin-letter>	パスワードに利用できる英文字を設定します。	A-Z a-z
	required 属性　true の場合は、少なくとも1文字は設定した文字が必要です。	false
<allow-number>	パスワードに利用できる数字を設定します。	0-9
	required 属性　true の場合は、少なくとも1文字は設定した文字が必要です。	false
<allow-extra-char>	パスワードに利用できるその他の文字を設定します。	_-.+$#!/@
	required 属性　true の場合は、少なくとも1文字は設定した文字が必要です。	false
<deny-old-password>	パスワード履歴に存在するパスワードの利用を禁止します。	false
<deny-userid>	ユーザ ID と同じパスワードの利用を禁止します。	false

*1　allow-latin-letter,allow-number,allow-extra-char の required 属性が true になっている数によって最小値の下限が変化します。3種類とも true の場合は、最小値の下限が3となります。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.4 二重ログイン防止機能

二重ログイン防止機能では、グループ管理者および一般ユーザに対し、同じユーザ ID のユーザが同時にログインして操作することを防ぐ機能を提供します。標準実装では以下の機能が利用できます。

二重ログイン防止機能の有効化

二重ログイン後にユーザ自身による自身の他アクティブセッション無効化機能の有効化

※上記機能は、グループ毎にグループ管理者および一般ユーザそれぞれに対し設定が可能です。

5.4.1 二重ログイン防止機能における注意事項

二重ログイン防止機能はグループ管理者および一般ユーザに対して利用できる機能です。システム管理者に対しては、二重ログイン防止機能はご利用できません。

ログアウトせずブラウザを閉じた場合、通常そのアクティブセッションがセッションタイムアウトするまでログインできません。これを回避する方法として、そのアクティブセッションを無効化する機能があります。

5.4.2 二重ログイン防止機能を利用した場合の画面遷移

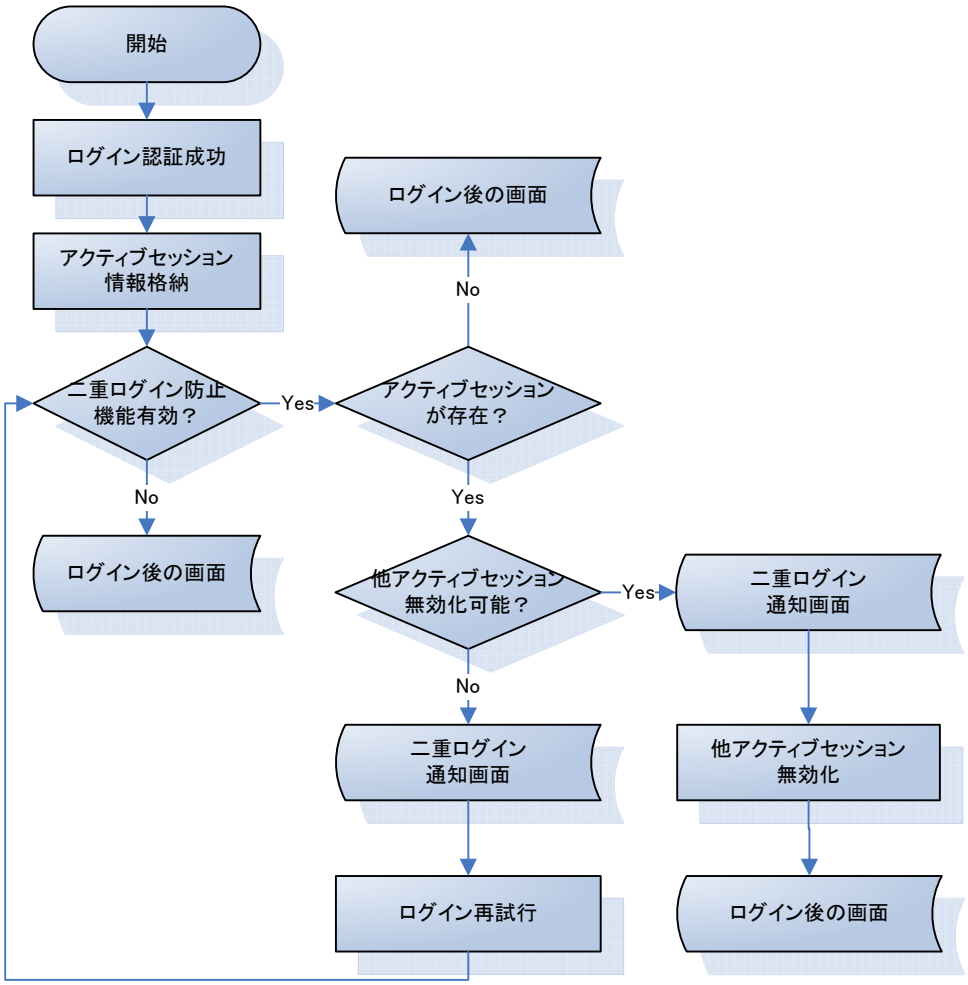


図 84 二重ログイン防止機能を利用した場合の画面遷移図

5.4.3 アクティブセッション

アクティブセッションとは、ログインしたセッションに関するセッション情報です。ログイン認証成功後に作成され、ユーザがログアウトするかセッションがタイムアウトした時点でアクティブセッション情報は削除されます。このアクティブセッション情報を利用してユーザのログイン状況を管理します。アクティブセッションマネージャは、アクティブセッション情報を管理するためのクラスであり、アクティブセッション情報を監視したり、アクティブセッションを強制的に無効化することができます。

- アクティブセッションにおける注意事項
  アクティブセッション情報はグループ管理者および一般ユーザに対して管理されます。システム管理者に対しては、アクティブセッション情報は管理されません。

- アクティブセッションを利用した場合の画面遷移

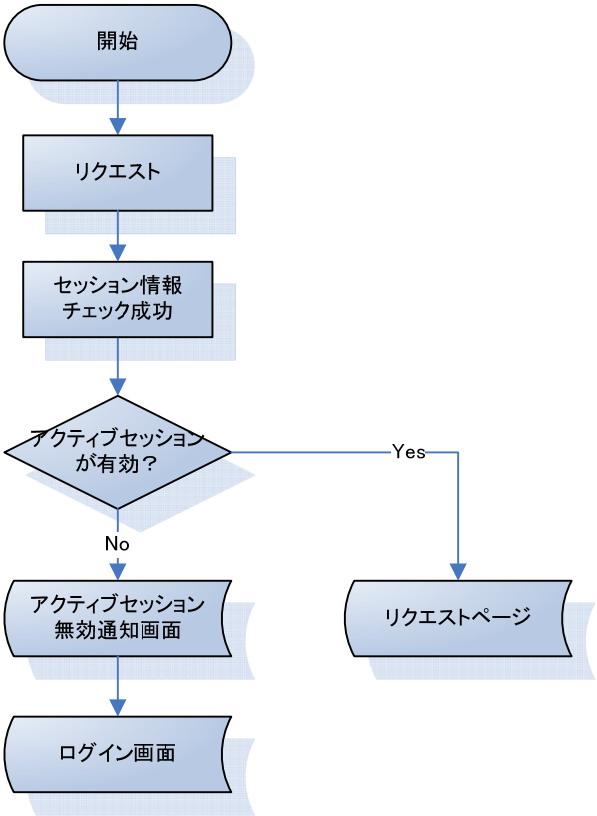


図 85 アクティブセッションを利用した場合の画面遷移図

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.4.4 アクティブセッションマネージャ

アクティブセッションマネージャでは、アクティブセッション情報の管理を行うマネージャクラスです。
スクリプト開発モデルでは ActiveSessionManager
JAVAEE 開発モデルでは jp.co.intra_mart.foundation.security.activesession.ActiveSessionManager
です。

5.4.4.1 アクティブセッション情報操作

アクティブセッションマネージャはアクティブセッション設定に従い、アクティブセッション操作モジュール(アクティブセッションアクセッサ)からアクティブセッション情報を操作します。

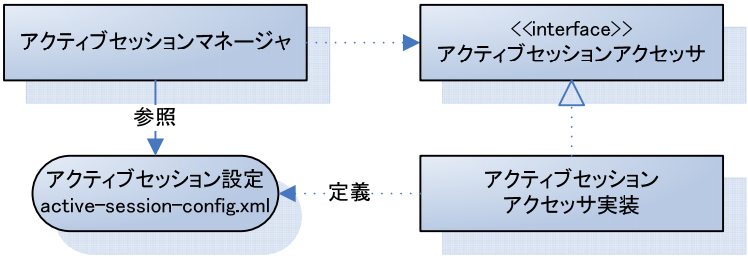


図 86 アクティブセッションマネージャのアーキテクチャ

5.4.4.2 アクティブセッション定義情報

アクティブセッション操作モジュールの設定を行います。
設定ファイルは、ServerManager/conf/active-session-config.xml に記述します。
設定例を以下に示します。
この設定を元にアクティブセッションマネージャはアクティブセッション操作モジュールの実装を取得します。

```
<active-session-config>

  <session-config>
    <active-session-binding-listener>my_listener</active-session-binding-listener>
  </session-config>

  <navigator-distinction default-app-name="Unknown">
    <navigator index-of-user-agent="MSIE">
      <app-name>Internet Explorer</app-name>
    </navigator>
    <navigator index-of-user-agent="Netscape">
      <app-name>Netscape</app-name>
    </navigator>
    ....
  </navigator-distinction>


  <group-default accessor-class="StandardActiveSessionAccessor">
    ....
  </group-default>

  <group name="sample" accessor-class="StandardActiveSessionAccessor">
    ....
  </group>

</active-session-config>
```

図 87 アクティブセッション情報設定例

■ タグの説明

<session-config>		セッションに保管する情報のパラメータ名を任意に変更できます。 変更したい項目だけ記述します。記述しない項目は、デフォルトのパラメータ名が使用されます。 通常、変更の必要はありません。
	<active-session-binding-listener>	アクティブなセッションを登録・削除するためのインスタンスをセッションに保管するためのパラメータ名を設定します。省略した場合は、「jp.co.intra_mart.session.active.session.binding.listener」となります。

<navigator-distinction>		ユーザエージェントからナビゲータ (Web ブラウザ) を判別する設定を記述します。 default-app-name 属性は、ナビゲータを判別で一致しなかった場合のナビゲータ名です。 通常、変更の必要はありません。
	<navigator>	ナビゲータ情報を設定します。(必須項目) これは、どのナビゲータからのアクティブセッションかを判別するために使用します。index-of-user-agent 属性で指定された値(文字列)がユーザエージェントに含まれる場合、そのアクティブセッションはそのナビゲータからアクセスしてきたと判別します。ナビゲータ判別は上から定義順に調べます。
		<app-name> ナビゲータ名を設定します。(必須項目)

<group-default>	各ログイングループのデフォルト設定を行います。(必須項目) group タグの name 属性に設定されたログイングループはその group タグ内の設定が優先されます。
	accessor-class 属性 ： アクティブセッション管理モジュールの実装クラスを設定します。 実装クラスは jp.co.intra_mart.foundation.security.activesession.ActiveSessionAccessor インタフェースを実装している必要があります。 省略した場合は、 jp.co.intra_mart.foundation.security.activesession.StandardActiveSessionAccessor を設定します。

<group>	ログイングループ単位の設定を行います。(任意項目)
	name 属性 ： 対象となるログイングループ ID を設定します。(必須項目)
	accessor-class 属性 ： アクティブセッション管理モジュールの実装クラスを設定します。 実装クラスは jp.co.intra_mart.foundation.security.activesession.ActiveSessionAccessor インタフェースを実装している必要があります。 省略した場合は、group-default タグの accessor-class 属性に設定したクラスが使用されます。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.4.4.3 アクティブセッション情報

アクティブセッション情報は、セッション ID と紐づくさまざまな情報を格納するオブジェクトです。

項目	説明	型	プロパティ
セッション ID	セッション(HttpSeesion)を特定する ID	String	sessionId
ログインタイプ	ログインユーザタイプ	String	loginType
ログイングループ	ログイングループ ID	String	loginGroup
ユーザ ID	ログインユーザ ID	String	userId
ログイン時刻	ログインした時刻 (GMT 1970 年 1 月 1 日 0 時からのミリ秒数)	Java ： long JavaScript ：Number	loginTime
リモートアドレス	クライアント IP アドレス	String	remortAddr
ユーザエージェント	ユーザエージェント(ブラウザ情報)	String	userAgent
最終応答時刻	最後の要求への応答時刻 (GMT 1970 年 1 月 1 日 0 時からのミリ秒数)	Java ： long JavaScript ：Number	lastResponseTime
セッションタイムアウト	セッションを保ち続ける最大の時間間隔 (秒数)	Java ： int JavaScript ：Number	expireInterval
処理中フラグ	要求への応答処理中かどうかのフラグ	boolean	processing
有効フラグ	アクティブセッションが有効化どうかのフラグ	boolean	valid
無効化ログインタイプ	アクティブセッションを無効にしたログインユーザタイプ	String	invalidateLoginType
無効化ユーザ ID	アクティブセッションを無効にしたログインユーザ ID	String	invalidateUserId
無効化時刻	アクティブセッションを無効化した時刻 (GMT 1970 年 1 月 1 日 0 時からのミリ秒数)	Java ： long JavaScript ：Number	invalidateTime

5.4.4.4 アクティブセッション情報のモデル

- **jp.co.intra_mart.foundation.security.activesession.ActiveSessionInfo** クラス

  アクティブセッション情報を格納するクラスです。

  アクティブセッション情報については、このクラスを通して、操作します。

5.4.4.5 アクティブセッション情報操作モジュールインタフェース

■ **jp.co.intra_mart.foundation.security.activesession.ActiveSessionAccessor インタフェース**

アクティブセッション情報の管理およびアクティブセッションの有効性チェックを行うためのインタフェースです。

5.4.4.6 アクティブセッション情報操作実装モジュール

■ **jp.co.intra_mart.foundation.security.activesession.StandardActiveSessionAccessor クラス**

標準のアクティブセッション情報の管理およびアクティブセッションの有効性チェックを行うための実装クラスです。
このクラス名は設定ファイル（ServerManager/conf/active-session-config.xml）に記述します。
設定するタグは、<group-default>の accessor-class 属性かまたは、<group>の accessor-class 属性に記述します。

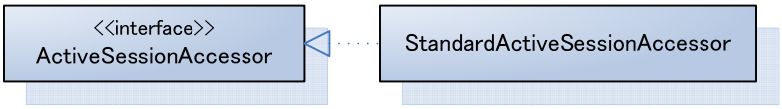


図 88 StandardActiveSessionAccessor クラス

◆ 初期化パラメータ

パラメータ実装例は以下の通りです。
<group-default>タグと<group>タグには同じ設定項目が存在します。
ただし、ログインしているログイングループと<group>タグの name 属性値が同じタグ内の設定が優先して使用されます。一致する<group>タグがないまたは、タグ内の項目が存在しない場合は、<group-default>タグの設定が利用されます。

```
<active-session-config>
  <session-config/>
  <navigator-distinction>
      ......
  </navigator-distinction>
  <group-default accessor-class="SampleStandardActiveSessionAccessor">
    <access-security-page-provider provider-class="AccessSecurityPageProvider">
      <access-security-page page-id="active-session-invalidated">
        <location request-header="maskat_layoutid">…</location>
        <location client-type="mobile">…</location>
        <location>…</location>
      </access-security-page>
    </access-security-page-provider>
    <active-session-invalidated-page-id>
      active-session-invalidated
    </active-session-invalidated-page-id>
  </group-default>
  <group name="default">
      ......
  </group>
</active-session-config>
```


Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

表 84 StandardActiveSessionAccessor クラスのパラメーター覧

タグ名	説明	デフォルト
<active-session-observer>	アクティブセッション管理に関する設定です。	なし
	enable 属性 アクティブセッション監視機能の有効/無効を設定します。**二重ログイン防止機能を利用する場合は必ず有効(true)にしてください。**	true
<access-security-page-provider>	アクティブセッション操作で使用するアクセスセキュリティページに関する設定です。	なし
	provider-class 属性 アクセスセキュリティページプロバイダ実装クラスを設定します(変更不可)	*1
<active-session-invalidated-page-id>	アクティブセッションが無効の場合に遷移するページ ID。指定するページ ID は、access-security-page-provider で定義されたアクセスセキュリティページでなければなりません。	*2
<session-id-manipulator>	セッション ID 操作機能を利用する際の設定です。	なし
	enable 属性 セッション ID 操作機能の有効/無効を設定します。セッション ID 操作機能を利用する場合は必ず有効(true)にしてください。	false

*1　デフォルトは jp.co.intra_mart.system.security.AccessSecurityPageProviderImpl です。

*2　デフォルトは active-session-invalidated です。

表 85 access-sesurity-page-provider タグ内のパラメーター覧

タグ名	説明	デフォルト
<access-security-page>	アクセスセキュリティページに関する設定です。	なし
	page-id 属性　アクセスセキュリティページの ID です。	*1

*1　アクティブセッション操作モジュールでのデフォルトは active-session-invalidated です。

表 86 access-sesurity-page タグ内のパラメーター覧

タグ名	説明	デフォルト
<location>	画面 URL を設定します。	*1
	request-header 属性　リクエストヘッダに指定されたリクエストヘッダ名が含まれている場合の画面 URL を設定します。	なし
	client-type 属性　クライアントタイプが指定された値の場合の画面 URL を設定します。	なし

*1　アクティブセッション操作モジュールでのデフォルトは system/security/error/active_session_invalidated_page.jssp です。

表 87 session-id-manipulator タグ内のパラメーター覧

タグ名	説明	デフォルト
<session-id-start-point>	セッション ID 操作での開始ポイントの設定です。 開始ポイントには、開始文字数を指定してください。	なし *1
<session-id-end-point>	セッション ID 操作での終了ポイントの設定です。 終了ポイントには、終了文字数を指定してください。	なし *2

*1、及び、*2 以下の場合、セッション ID 操作機能は無効となります。

- 正整数(0 を除く)以外の文字列の設定
- 無設定
- 「*1 設定値」 >「*2 設定値」
- セッション ID 長以上の値の設定

■ 開始ポイント と 終了ポイントの設定について

「**ABCDEF**」というセッション ID があった場合、文字数は以下のようになります。

1 文字目 → A
2 文字目 → B
3 文字目 → C
4 文字目 → D
5 文字目 → E
6 文字目 → F

開始ポイントに「**2**」、終了ポイントに「**5**」を指定した場合、「**BCDE**」の文字列がアクティブセッション情報のセッション ID として登録されます。

■ セッション ID 操作機能の制限事項について

セッション ID 操作機能を利用するには、設定ファイル（ServerManager/conf/active-session-config.xml）の編集が必要であると共に、 intra-mart の再起動が必要となります。また、本機能を有効とした際は、あらかじめ、「B_M_ACTIVE_SESSION_INFO」テーブルの全レコードを削除してください。

セッション ID 操作機能無効時のデータが残っていますと、予期せぬエラーが発生し、二重ログイン防止機能が正常に動作しません。

エラーが発生する操作は以下の手順となります。

① 2 重ログイン防止機能を有効、セッション ID 操作機能を無効とし、intra-mart を起動する。
② ブラウザ A でログインする。(ブラウザ A のセッション ID は操作されずに登録が行われる)
③ セッション ID 操作機能を有効とし、intra-mart を再起動する。
④ ブラウザ B でログインする。(ブラウザ B のセッション ID は操作され登録が行われる)
⑤ ブラウザ B で 2 重ログイン検出画面の「セッションを無効」ボタンを押下する。

手順⑤の「セッションを無効」ボタンを押下する際、ブラウザ A のセッションを無効とする処理を行いますが、セッション無効処理内でブラウザ A のセッション ID が操作されるため、手順②で登録しているセッション ID と異なり、無効とするセッションが見つからないことで例外が発生します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.4.5 二重ログインマネージャ

二重ログインマネージャは、二重ログインのチェックおよび、二重ログインに関する設定を管理するマネージャクラスです。

スクリプト開発モデルでは DuplicateLoginManager

JAVAEE 開発モデルでは jp.co.intra_mart.foundation.security.duplication.DuplicateLoginManager

です。

二重ログインマネージャは、アクティブセッションマネージャを使用して二重ログインを管理します。具体的には、各ユーザのアクティブセッション数からログイン数を調べ、二重ログインを管理します。

5.4.5.1 二重ログイン防止設定取得

二重ログイン管理マネージャは二重ログイン防止設定に従ってその振る舞いが決定されます。

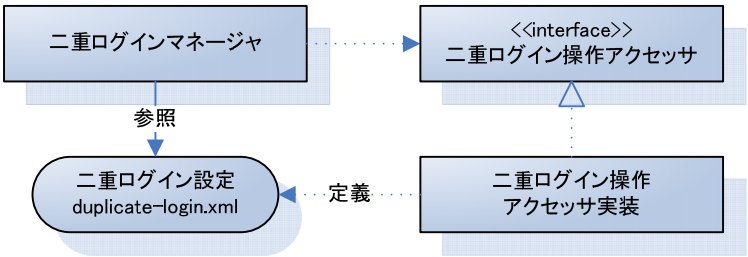


図 89 二重ログインマネージャのアーキテクチャ

5.4.5.2 二重ログイン防止定義情報

二重ログイン操作モジュールの設定を行います。

設定ファイルは、ServerManager/conf/duplicate-login.xml に記述します。

設定例を以下に示します。

この設定を元に二重ログインマネージャは二重ログイン操作モジュールの実装を取得します。

```
<duplicate-login>

  <request-config>
    <requested-page>jp.co.intra_mart.request.duplicate.login.requested.page</requested-page>
    <invalidate-other-sessions>
      jp.co.intra_mart.request.duplicate.login.invalidate.other.sessions
    </invalidate-other-sessions>
  </request-config>

  <group-default accessor-class="StandardDuplicateLoginAccessor">
    ....
  </group-default>

  <group name="sample" accessor-class="StandardDuplicateLoginAccessor">
    ....
  </group>

</duplicate-login>
```

図 90 二重ログイン防止情報設定例

Re

■ タグの説明

<request-config>	リクエスト属性に付加する情報のパラメータ名を任意に変更できます。 変更したい項目だけ記述します。記述しない項目は、デフォルトのパラメータ名が使用されます。 通常、変更の必要はありません。	
	<requested-page>	リクエストされた画面 URL を二重ログイン検出画面に渡すために利用するリクエスト属性名を設定します。 省略した場合、 「jp.co.intra_mart.request.duplicate.login.requested.page」となります。
	<invalidate-other-sessions>	二重ログイン検出画面から、ユーザ自身による自身の他アクティブセッションを無効化する際に利用するリクエスト属性名を設定します。 省略した場合、 「jp.co.intra_mart.request.duplicate.login.invalidate.other.sessions」となります。

<group-default>	各ログイングループのデフォルト設定を行います。(必須項目) group タグの name 属性に設定されたログイングループはその group タグ内の設定が優先されます。
	accessor-class 属性 :二重ログイン操作モジュールの実装クラスを設定します。 実装クラスは jp.co.intra_mart.foundation.security.duplication.DuplicateLoginAccessor インタフェースを実装している必要があります。

<group>	ログイングループ単位の設定を行います。(任意項目)
	name 属性 : 対象となるログイングループ ID を設定します。(必須項目)
	accessor-class 属性 : 二重ログイン操作モジュールの実装クラスを設定します。 実装クラスは jp.co.intra_mart.foundation.security.duplication.DuplicateLoginAccessor インタフェースを実装している必要があります。 省略した場合は、 jp.co.intra_mart.foundation.security.duplication.StandardDuplicateLoginAccessor を設定します。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.4.5.3 二重ログイン操作モジュールインタフェース

■ **jp.co.intra_mart.foundation.security.duplication.DuplicateLoginAccessor インタフェース**

二重ログインに関する管理および二重ログインチェックを行うためのインタフェースです。

5.4.5.4 二重ログイン操作実装モジュール

■ **jp.co.intra_mart.foundation.security.duplication.StandardDuplicationLoginAccessor クラス**

標準の二重ログインに関する管理および二重ログインチェックを行うための実装クラスです。このクラス名は設定ファイル（ServerManager/conf/duplicate-login.xml）に記述します。

設定するタグは、<group-default>の accessor-class 属性かまたは、<group>の accessor-class 属性に記述します。

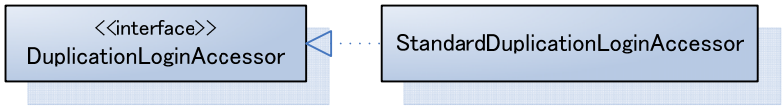


図 91 StandardDuplicateLoginAccessor クラス

◆ 初期化パラメータ

パラメータ実装例は以下の通りです。

<group-default>タグと<group>タグには同じ設定項目が存在します。

ただし、ログインしているログイングループと<group>タグの name 属性値が同じタグ内の設定が優先して使用されます。

一致する<group>タグがないまたは、タグ内の項目が存在しない場合は、<group-default>タグの設定が利用されます。

```
<duplicate-login>
  <request-config/>
  <group-default accessor-class="SampleStandardDuplicateLoginAccessor">
    <access-security-page-provider provider-class="AccessSecurityPageProvider">
      <access-security-page page-id="duplicate-login-detection">
        <location request-header="maskat_layoutid">…</location>
        <location client-type="mobile">…</location>
        <location>…</location>
      </access-security-page>
    </access-security-page-provider>
    <duplicate-login-detection-page-id>
      duplicate-login-detection
    </duplicate-login-detection-page-id>
    <group-duplicate-login-detect enable="false">
      <invalidatable-by-user>false</invalidatable-by-user>
    </group-duplicate-login-detect>
    <duplicate-login-detect enable="false">
      <invalidatable-by-user>false</invalidatable-by-user>
    </duplicate-login-detect>
  </group-default>
  <group name="default">
      ……
  </group>
</duplicate-login>
```

表 88 StandardDuplicateLoginAccessor クラスのパラメーター覧

タグ名	説明	デフォルト
<access-security-page-provider>	二重ログイン操作で使用するアクセスセキュリティページに関する設定です。	なし
	provider-class 属性 アクセスセキュリティページプロバイダ実装クラスを設定します(変更不可)	*1
<duplicate-login-detection-page-id>	二重ログインが検出された場合に遷移するページ ID。 指定するページ ID は、access-security-page-provider で定義されたアクセスセキュリティページでなければなりません。	*2
<group-duplicate-login-detect>	グループ管理者に対する二重ログインの設定です。	なし
	enable 属性 二重ログイン防止を行うかどうかを設定します。	false
<duplicate-login-detect>	一般ユーザに対する二重ログインの設定です。	なし
	enable 属性 二重ログイン防止を行うかどうかを設定します。	false

*1　デフォルトは jp.co.intra_mart.system.security.AccessSecurityPageProviderImpl です。

*2　デフォルトは duplicate-login-detection です。

表 89 access-sesurity-page-provider タグ内のパラメーター覧

タグ名	説明	デフォルト
<access-security-page>	アクセスセキュリティページに関する設定です。	なし
	page-id 属性　アクセスセキュリティページの ID です。	*1

*1　二重ログイン操作モジュールでのデフォルトは duplicate-login-detection です。

表 90 access-sesurity-page タグ内のパラメーター覧

タグ名	説明	デフォルト
<location>	画面 URL を設定します。	*1
	request-header 属性　リクエストヘッダに指定されたリクエストヘッダ名が含まれている場合の画面 URL を設定します。	なし
	client-type 属性　クライアントタイプが指定された値の場合の画面 URL を設定します。	なし

*1　二重ログイン操作モジュールでのデフォルトは

system/security/error/duplicate_login_detection_page.jssp です。

表 91 group-duplicate-login-detect タグおよび duplicate-login-detect タグ内のパラメーター覧

タグ名	説明	デフォルト
<invalidatable-by-user>	ユーザ自身で自身のセッションの無効化を行えるか設定します。	false

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.4.6 [参考] ログイン中にログイン画面へのアクセスを防ぐ方法

ログイン中にログイン画面を表示しようとした際に警告ページを表示するためのフィルタを Ver.7.0.5 で追加しました。このフィルタは、クッキーにセッション ID が残っている状態でログイン画面がリクエストされたときに、そのセッション ID がログインしている状態の場合、警告ページを表示します。
このフィルタを設定することで、システム利用者が「リリースノート 6 - 制限事項 - 6.9 ブラウザ(5)」となるオペレーションを実行したときに警告を表示することができるようになります。

5.4.6.1 フィルタInitialPageSessionHandlingFilterの設定方法

このフィルタは以下のサーブレットに対して使用されることを想定しています。
UserInitialServlet
GroupSuperUserInitialServlet
SuperUserInitialServlet
この機能を利用する場合、そのサーブレットのフィルタマッピングの設定の最後に追加してください。以下は、UserInitialServlet に本フィルタを設定する場合の設定例です。

```
<filter>
    <filter-name>InitialPageSessionHandlingFilter</filter-name>
    <filter-class>jp.co.intra_mart.foundation.security.filter.InitialPageSessionHandlingFilter</filter-class>
    <init-param>
        <param-name>enable</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>warning-page-path</param-name>
        <param-value>system/security/error/request_session_already_logon_error.jssp</param-value>
    </init-param>
</filter>

    : (省略)

<filter-mapping>
    <filter-name>InitialPageSessionHandlingFilter</filter-name>
     <servlet-name>UserInitialServlet</servlet-name>
</filter-mapping>
```

5.5 暗号化モジュール

暗号化モジュールでは、さまざまな暗号化アルゴリズムを利用できる構造を提供します。
ユーザ独自の暗号化アルゴリズムを作成して、システムに組み込むことが可能です。

暗号化モジュールを利用する関数は AccessSecurityManager の以下の関数です。

encrypt([String category],String plane)	指定したカテゴリの暗号化アルゴリズムを使用して、暗号化を行います。 カテゴリを省略した場合、「default」のカテゴリを指定した場合と同じになります。 また、「default」のカテゴリが存在しない場合は、jp.co.intra_mart.foundation.security.cryption.StandardCryption が使用されます。
decrypt([String category],String secret)	指定したカテゴリの暗号化アルゴリズムを使用して、複合化を行います。 カテゴリを省略した場合、「default」のカテゴリを指定した場合と同じになります。 また、「default」のカテゴリが存在しない場合は、jp.co.intra_mart.foundation.security.cryption.StandardCryption が使用されます。

5.5.1 暗号化アルゴリズムの追加

暗号化アルゴリズムを追加するには、サーバマネージャ/conf/cryption-config.xml を編集します。

```
<cryption-config>
  <cryption category="default">
    <cryption-class>jp.co.intra_mart.foundation.security.cryption.StandardCryption</cryption-class>
  </cryption>
  <cryption category="mobile-user-agent">
    <cryption-class>jp.co.intra_mart.foundation.security.cryption.StandardCryption</cryption-class>
  </cryption>
</cryption-config>
```

■ タグの説明

<cryption>		暗号化アルゴリズムの設定を行います。 cryption タグの category 属性に暗号化アルゴリズムカテゴリ名を設定します。(必須)
	<cryption-class>	暗号化アルゴリズムの実装クラスを設定します。(必須) 実装クラスは jp.co.intra_mart.foundation.security.cryption.StandardCryption インタフェースを実装している必要があります。

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.6 汎用新着ポートレット連携

パスワード履歴管理では、パスワード有効期限切れの接近を汎用新着ポートレットに表示することが可能です。
パスワード有効期限切れの通知は、ユーザ属性によってユーザ個々に何日前からポートレットに表示するかを設定抱きます。

5.6.1 汎用新着ポートレット定義

標準では、パスワード有効期限切れ通知用の汎用新着ポートレット定義ファイルは、
StorageServer/portal/system_notice.xml に記述されています。
汎用新着ポートレットの登録方法は、ポータル仕様書を参照してください。

```
<new-arrived-portlet>
  <provider>
    <provider-class>
      jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider
    </provider-class>
  </provider>

</new-arrived-portlet>
```

5.6.2 パスワード有効期限通知の実装クラス

■ **jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider クラス**
汎用新着ポートレットに対して、パスワード期限切れの接近を通知するプロバイダークラスです。

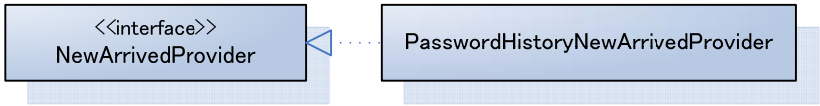


図 92 PasswordHistoryNewArrivedProvider クラス

◆ 初期化パラメータ
デフォルト値をパラメータとして設定可能です。

表 92 PasswordHistoryNewArrivedProvider クラスのパラメータの一覧

パラメータ名	説明
report-limit-days	何日前からパスワードの期限切れを通知するかのデフォルト値を設定します。 デフォルト値は5日前です
link-page	リンク先のページです。 スクリプト開発モデルのページ URL の場合は拡張子(.jssp)を付加します。 デフォルト値はパスワード変更画面 /system/setting/user/password/main.jssp

5.7 モバイル端末個体識別番号取得モジュール

モバイル端末からアクセスする場合、その端末から個体識別番号を取得する機能です。
個体識別番号は、各キャリアにより提供方法が異なります。
各キャリア単位で解析モジュールを用意して、個体識別番号を取り出します。

5.7.1 モバイル端末個体識別番号取得メソッド

モバイル端末個体識別番号取得メソッドでは、取得できた個体識別番号を返します。
個体識別番号が取得できなかった場合は、null を返します。
モバイル端末個体識別番号取得メソッドは以下の通りです。

開発モデル	モバイル端末個体識別番号取得メソッド
JavaEE 開発モデル	jp.co.intra_mart.foundation.security.AccessSecurityManager クラス getMobileUserAgent() getMobileUserAgent(HttpServletRequest request, HttpServletResponse response)
Script 開発モデル	AccessSecurityManager オブジェクト getMobileUserAgent()

5.7.2 モバイル端末個体識別番号取得アーキテクチャ

モバイル端末個体識別番号取得を取得する場合、各キャリア単位にモバイル端末個体識別番号取得モジュールを作成し、そのモジュールを設定ファイルに記述することで、取得できるようにしています。

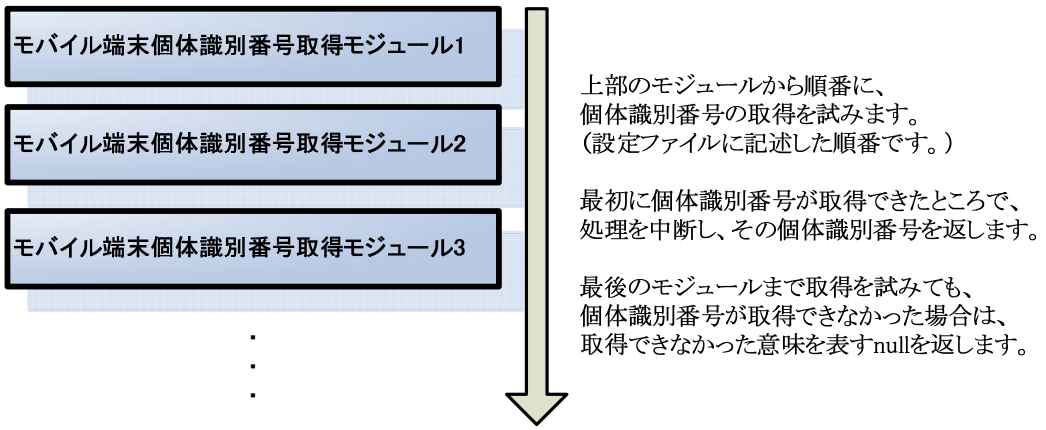


図 93 モバイル端末個体識別番号取得アーキテクチャ

Copyright 2000-2013 株式会社 NTT データ イントラマート All rights Reserved.

5.7.3 モバイル端末個体識別番号取得設定ファイル

設定ファイルは、[サーバマネージャをインストールしたディレクトリ]/conf/mobile-user-agent.xml です。

モバイル端末個体識別番号取得モジュールを試みたい順番に記述します。

```
<mobile-user-agent>

  <user-agent>
    <provider-class>jp.co.intra_mart.foundation.security.mobile.MobileEasyLoginDocomoAnalyzer</provider-class>
  </user-agent>

  <user-agent>
    <provider-class>jp.co.intra_mart.foundation.security.mobile.MobileEasyLoginAuAnalyzer</provider-class>
  </user-agent>
      ・
      ・
      ・
</mobile-user-agent>
```

■ タグの説明

<user-agent>		モバイル端末個体識別番号取得モジュールの設定を行います。(複数設定可能)
	<provider-class>	モバイル端末個体識別番号取得モジュールの実装クラスを設定します。(必須) jp.co.intra_mart.foundation.security.mobile.MobileUserAgentProvider インタフェースを実装している必要があります。 この実装クラスが作成された時、一度だけ init()メソッドが呼び出されます。 このメソッドの引数には、user-agent タグのノード(Node)が渡されます。 user-agent タグ内に、任意の初期化パラメータ用のタグを作成することが可能です。

5.7.4 モバイル端末個体識別番号取得プロバイダインタフェース

■ **jp.co.intra_mart.foundation.security.mobile.MobileUserAgentProvider** インタフェース

モバイル端末個体識別番号取得するためのインタフェースです。

モバイル端末個体識別番号取得モジュールを作成する場合は、このインタフェースを実装する必要があります。

5.7.5 モバイル端末個体識別番号取得モジュール実装クラス

いくつかのキャリアに対してのモバイル端末個体識別番号取得モジュールが標準で用意されています。

この実装クラス群のパッケージは、jp.co.intra_mart.foundation.security.mobile です。

キャリア (方式)	モバイル端末個体識別番号取得モジュール実装クラス
Au 用 (WAP2.0)	MobileEasyLoginAuAnalyzer
Au 用 (HDML)	MobileEasyLoginAuHdmlAnalyzer
NTT DoCoMo 用 (FOMA)	MobileEasyLoginDocomoAnalyzer
NTT DoCoMo 用 (move)	MobileEasyLoginDocomoMoveAnalyzer
J-PHONE 用	MobileEasyLoginJphoneAnalyzer
SoftBank 用	MobileEasyLoginSoftBankAnalyzer
Vordafone 用	MobileEasyLoginVodafoneAnalyzer

intra-mart WebPlatform/AppFramework Ver.7.0
アクセスセキュリティ仕様書

2013/10/18 第9版

Copyright 2000-2013 株式会社NTTデータ イントラマート
All rights Reserved.
TEL: 03-5549-2821
FAX: 03-5549-2816
E-MAIL: info@intra-mart.jp
URL: http://www.intra-mart.jp/