Oracle® Application Server Single Sign-On

アプリケーション開発者ガイド

10*g*（9.0.4）

部品番号：B12373-01

2004 年 3 月

ORACLE®

Oracle Application Server Single Sign-On アプリケーション開発者ガイド , 10*g*（9.0.4）

部品番号：B12373-01

原本名：Oracle Application Server Single Sign-On Application Developer's Guide, 10*g* (9.0.4)

原本部品番号：B10852-01

原本著者：Henry Abrecht

原本協力者：Kamalendu Biswas, Ganesh Kirti, Arun Swaminathan

目次

はじめに vii

対象読者 viii
このマニュアルの構成 viii
関連ドキュメント ix
表記規則 ix

1 はじめに

2 シングル・サインオン対応のアプリケーションの開発

mod_osso を使用したアプリケーションの保護 : 2 つの方法 2-2
URL の静的な保護 2-2
ダイナミック・ディレクティブを使用した URL の保護 2-3
mod_osso を使用したアプリケーションの開発 2-3
静的に保護された PL/SQL アプリケーションの開発 2-4
静的に保護された Java アプリケーションの開発 2-6
ダイナミック・ディレクティブを使用する Java アプリケーションの開発 2-7
Java の例 1: 簡易認証 2-8
Java の例 2: シングル・サインオフ 2-9
Java の例 3: 強制認証 2-10
非 GET 認証について 2-11
セキュリティに関する問題 : シングル・サインオフとアプリケーションのログアウト 2-12
アプリケーションのログイン : コード例 2-12
不適切なコード例 1 2-12
不適切なコード例 2 2-13
推奨コード 2-13

アプリケーションのログアウト : 推奨コード 2-14

A Single Sign-On Software Development Kit

PL/SQL API A-2
ファンクションとプロシージャ A-2
GENERATE_REDIRECT ファンクション A-2
構文 A-2
例 A-3
PARSE_URL_COOKIE プロシージャ A-4
構文 A-4
例 A-5
GET_ENABLER_CONFIG プロシージャ A-5
構文 A-5
例 A-6
CREATE_ENABLER_CONFIG プロシージャ A-6
構文 A-6
例 A-7
UPDATE_ENABLER_CONFIG プロシージャ A-8
構文 A-8
例 A-9
DELETE_ENABLER_CONFIG プロシージャ A-9
構文 A-9
例 A-9
ENCRYPT_COOKIE ファンクション A-10
構文 A-10
例 A-10
DECRYPT_COOKIE ファンクション A-10
構文 A-10
例 A-11
表の定義 A-11
WWSEC_ENABLER_CONFIG_INFO$ A-11
WWSEC_SSO_LOG$ A-11
例外 A-12
Java API A-12

B PL/SQL API および Java API の使用方法

API を使用する前に B-2

PL/SQL API を使用したパートナ・アプリケーションの作成 B-2
SAMPLE_SSO_PAPP.SSOAPP B-2
SAMPLE_SSO_PAPP.SIGN_ON B-2
SAMPLE_SSO_PAPP.LOGOUT B-2
Java API を使用したパートナ・アプリケーションの作成 B-3
サーブレットのパートナ・アプリケーション B-3
JSP のパートナ・アプリケーション B-4

C　SDK を使用したアプリケーションの追加と編集

「パートナ・アプリケーションの追加」ページ C-2
「パートナ・アプリケーションの編集」ページ C-3

D　パートナ・アプリケーションに渡されるユーザー属性

用語集

索引

表リスト

2-1 一般的に要求されるダイナミック・ディレクティブ 2-3
A-1 GENERATE_REDIRECT のパラメータ A-3
A-2 PARSE_URL_COOKIE のパラメータ A-4
A-3 GET_ENABLER_CONFIG のパラメータ A-5
A-4 CREATE_ENABLER_CONFIG のパラメータ A-7
A-5 UPDATE_ENABLER_CONFIG のパラメータ A-8
A-6 DELETE_ENABLER_CONFIG のパラメータ A-9
A-7 ENCRYPT_COOKIE のパラメータ A-10
A-8 DECRYPT_COOKIE のパラメータ A-10
A-9 例外 A-12
C-1 パートナ・アプリケーションへのログイン C-2
C-2 ログイン可能期間 C-2
C-3 アプリケーション管理者 C-3
C-4 「パートナ・アプリケーションの編集」ページのフィールド C-3
D-1 パートナ・アプリケーションに渡されるユーザー属性 D-2

はじめに

『Oracle Application Server Single Sign-On アプリケーション開発者ガイド』は、OracleAS Single Sign-On 用アプリケーションの変更を行う開発者の方を対象にしています。この変更には、Oracle HTTP Server 上の認証モジュールである mod_osso か、Single Sign-On SDK のいずれかを使用します。このマニュアルは、UNIX および Windows NT/2000 プラットフォームを対象にしています。

注意： このマニュアルでは、シングル・サインオン・ファイルを参照する際に、UNIX 表記を使用していますが、`ssocfg` スクリプト以外のファイルの名前および場所は、UNIX と Windows で共通しています。Windows 版のシングル・サインオン・ファイルにアクセスするには、次のディレクトリに移動します。

`%ORACLE_HOME%¥`*`directory_path`*`¥`

「はじめに」の項目は次のとおりです。

- 対象読者
- このマニュアルの構成
- 関連ドキュメント
- 表記規則

対象読者

このマニュアルは、次の知識または能力を前提としています。

- 動作中の OracleAS にアクセスできるか、新しくインストールできること
- OracleAS の概念を理解していること
- PL/SQL または Java プログラミング言語に習熟していること

このマニュアルの構成

『Oracle Application Server Single Sign-On アプリケーション開発者ガイド』では、Oracle HTTP の認証モジュール mod_osso を使用してアプリケーションをシングル・サインオン対応にする方法について説明します。付録では、同じ目的に Single Sign-On SDK を使用する方法について説明します。

第 1 章「はじめに」

mod_osso と Single Sign-On SDK を紹介します。その他の Single Sign-On コンポーネントについても簡単に説明します。

第 2 章「シングル・サインオン対応のアプリケーションの開発」

HTTP 認証モジュール mod_osso が、どのように Oracle Single Sign-On 対応のアプリケーションを保護するかについて説明します。アプリケーションと mod_osso の統合をデモンストレーションするコードを紹介します。

付録 A「Single Sign-On Software Development Kit」

シングル・サインオン対応のアプリケーション向け PL/SQL API の一覧を示し、説明します。SDK には Java API も含まれています。

付録 B「PL/SQL API および Java API の使用方法」

PL/SQL および Java を使用したパートナ・アプリケーションの作成方法について説明します。両方の言語のコード例を紹介します。

付録 C「SDK を使用したアプリケーションの追加と編集」

SDK に統合されたアプリケーションを Single Sign-On Server へ追加または登録する方法について説明します。既存アプリケーションのレジストリの編集方法について説明します。

付録 D「パートナ・アプリケーションに渡されるユーザー属性」
Single Sign-On Server が検証また取得する Oracle Internet Directory のユーザー属性の一覧を示し、説明します。これらの属性は、パートナ・アプリケーションに渡す URLC トークンを構成するために使用します。

用語集
このマニュアルで使用する用語を定義します。

関連ドキュメント

詳細は、次の Oracle ドキュメントを参照してください。

- 『Oracle Application Server Single Sign-On 管理者ガイド』
- 『Oracle Application Server Single Sign-On API Reference』

リリース・ノート、インストール関連ドキュメント、ホワイト・ペーパーまたはその他の関連ドキュメントは、OTN-J（Oracle Technology Network Japan）から、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。登録は、次の Web サイトから無償で行えます。

`http://otn.oracle.co.jp/membership/`

すでに OTN-J のユーザー名およびパスワードを取得している場合は、次の URL で OTN-J Web サイトのドキュメントのセクションに直接接続できます。

`http://otn.oracle.co.jp/document/`

表記規則

この項では、このマニュアルの本文およびコード例で使用される表記規則について説明します。この項の内容は次のとおりです。

- 本文の表記規則
- コード例の表記規則
- Microsoft Windows オペレーティング・システム環境での表記規則

本文の表記規則

本文では、特定の項目が一目でわかるように、次の表記規則を使用します。次の表に、その規則と使用例を示します。

規則	意味	例
太字	太字は、本文中で定義されている用語および用語集に記載されている用語を示します。	この句を指定すると、**索引構成表**が作成されます。
固定幅フォントの大文字	固定幅フォントの大文字は、システム指定の要素を示します。このような要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージおよびメソッドがあります。また、システム指定の列名、データベース・オブジェクト、データベース構造、ユーザー名およびロールも含まれます。	NUMBER 列に対してのみ、この句を指定できます。 BACKUP コマンドを使用して、データベースのバックアップを作成できます。 USER_TABLES データ・ディクショナリ・ビュー内の TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびユーザーが指定する要素のサンプルを示します。このような要素には、コンピュータ名およびデータベース名、ネット・サービス名および接続識別子があります。また、ユーザーが指定するデータベース・オブジェクトとデータベース構造、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニットおよびパラメータ値も含まれます。	sqlplus と入力して、SQL*Plus をオープンします。 パスワードは、orapwd ファイルで指定します。 /disk1/oracle/dbs ディレクトリ内のデータ・ファイルおよび制御ファイルのバックアップを作成します。 hr.departments 表には、department_id、department_name および location_id 列があります。 QUERY_REWRITE_ENABLED 初期化パラメータを true に設定します。 oe ユーザーとして接続します。
	注意 : プログラム要素には、大文字と小文字を組み合せて使用するものもあります。これらの要素は、記載されているとおりに入力してください。	JRepUtil クラスが次のメソッドを実装します。
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。ここで、*old_release* とはアップグレード前にインストールしたリリースを示します。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus または他のコマンドライン文の例です。次のように固定幅フォントで表示され、通常のテキストと区別されます。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表に、コード例で使用される表記規則とその使用例を示します。

規則	意味	例
[ ]	大カッコは、カッコ内の項目を任意に選択することを表します。大カッコは、入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコは、カッコ内の項目のうち、1 つが必須であることを表します。中カッコは、入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択項目の区切りに使用します。項目のうちの 1 つを入力します。縦線は、入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のいずれかを示します。	
	■ 例に直接関連しないコードの一部が省略されている。	CREATE TABLE ... AS *subquery*;
	■ コードの一部を繰り返すことができる。	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関連しない複数の行が省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
その他の記号	大カッコ、中カッコ、縦線および省略記号以外の記号は、記載されているとおりに入力する必要があります。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック体	イタリック体は、特定の値を指定する必要があるプレースホルダや変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

規則	意味	例

大文字	大文字は、システム指定の要素を示します。これらの要素は、ユーザー定義の要素と区別するために大文字で示されます。大カッコ内にないかぎり、表示されているとおりの順序および綴りで入力します。ただし、大 / 小文字が区別されないため、小文字でも入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名などです。	`SELECT last_name, employee_id FROM employees;` `sqlplus hr/hr`
	注意 : プログラム要素には、大文字と小文字を組み合せて使用するものもあります。これらの要素は、記載されているとおりに入力してください。	`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Microsoft Windows オペレーティング・システム環境での表記規則

次の表に、Microsoft Windows オペレーティング・システム環境での表記規則とその使用例を示します。

規則	意味	例
ファイル名およびディレクトリ名	ファイル名およびディレクトリ名は大 / 小文字が区別されません。特殊文字の左山カッコ（<）、右山カッコ（>）、コロン（:）、二重引用符（"）、スラッシュ（/）、縦線（|）およびハイフン（-）は使用できません。円記号（¥）は、引用符で囲まれている場合でも、要素のセパレータとして処理されます。Windows では、ファイル名が ¥¥ で始まる場合、汎用命名規則が使用されていると解釈されます。	`c:¥winnt"¥"system32` は `C:¥WINNT¥SYSTEM32` と同じです。
Windows コマンド・プロンプト	Windows コマンド・プロンプトには、カレント・ディレクトリが表示されます。このマニュアルでは、コマンド・プロンプトと呼びます。コマンド・プロンプトのエスケープ文字はカレット（^）です。	`C:¥oracle¥oradata>`

規則	意味	例

特殊文字	Windows コマンド・プロンプトで二重引用符（"）のエスケープ文字として円記号（¥）が必要な場合があります。丸カッコおよび一重引用符（'）にはエスケープ文字は必要ありません。エスケープ文字および特殊文字の詳細は、Windows オペレーティング・システムのドキュメントを参照してください。	`C:¥>exp scott/tiger TABLES=emp QUERY=¥"WHERE job='SALESMAN' and sal<1600¥"`
		`C:¥>imp SYSTEM/`*`password`*
		`FROMUSER=scott TABLES=(emp, dept)`

HOME_NAME	Oracle ホームの名前を表します。ホーム名には、英数字で 16 文字まで使用できます。ホーム名に使用可能な特殊文字は、アンダースコアのみです。	`C:¥> net start Oracle`*`HOME_NAME`*`TNSListener`
ORACLE_HOME および *ORACLE_BASE*	Oracle8*i* より前のリリースでは、Oracle コンポーネントをインストールすると、すべてのサブディレクトリが最上位の *ORACLE_HOME* の直下に置かれました。*ORACLE_HOME* ディレクトリの名前は、デフォルトでは次のいずれかです。 ■ `C:¥orant`（Windows NT の場合） ■ `C:¥orawin98`（Windows 98 の場合） このリリースは、Optimal Flexible Architecture（OFA）のガイドラインに準拠しています。*ORACLE_HOME* ディレクトリ下に配置されないサブディレクトリもあります。最上位のディレクトリは *ORACLE_BASE* と呼ばれ、デフォルトでは `C:¥oracle` です。他の Oracle ソフトウェアがインストールされていないコンピュータに最新リリースの Oracle をインストールした場合、Oracle ホーム・ディレクトリは、デフォルトで `C:¥oracle¥ora90` に設定されます。Oracle ホーム・ディレクトリは、*ORACLE_BASE* の直下に配置されます。 このマニュアルに示すディレクトリ・パスの例は、すべて OFA の表記規則に準拠しています。	`%ORACLE_HOME%¥rdbms¥admin` ディレクトリへ移動します。

1
はじめに

OracleAS Single Sign-On は、Oracle Application Server（OracleAS）のコンポーネントです。OracleAS Single Sign-On により、ユーザーは 1 つのユーザー名とパスワードを使用して、OracleAS の補完製品のすべての機能や他の Web アプリケーションにログインできます。

OracleAS Single Sign-On は、次のコンポーネントから構成されます。

- OracleAS Single Sign-On Server

  ユーザーが経費報告、メール、福利厚生情報などのシングル・サインオン対応のアプリケーションに安全にログインできるようにするプログラム・ロジック。

- パートナ・アプリケーション

  Single Sign-On Server に認証機能を委任する OracleAS アプリケーション。

- 外部アプリケーション

  Single Sign-On Server に認証機能を委任しない Web アプリケーション。そのかわりに、これらのアプリケーションは HTML ログイン・フォームを表示して、アプリケーションにおけるユーザー名とパスワードを問い合せます。

関連項目：『Oracle Application Server Single Sign-On 管理者ガイド』の第 1 章「コンポーネントとプロセス : 概要」

OracleAS リリース 9.0.4 では、Oracle HTTP Server の認証モジュールである mod_osso を使用して、アプリケーションのシングル・サインオン機能を有効にします。mod_osso は、以前のリリースでパートナ・アプリケーションとの統合に使用されていた Single Sign-On SDK にかわる簡単な方法です。mod_osso は、Single Sign-On Server の単独のパートナ・アプリケーションとして機能することで、認証プロセスを単純化します。これにより、OracleAS アプリケーションでの透過的な認証が実現します。アプリケーション管理者にとっては、Single Sign-On SDK と OracleAS アプリケーションの統合という負担がなくなります。

SDK は推奨されていません。リリース 9.0.2 の SDK を使用してアプリケーションを構築した場合、オラクル社では、mod_osso を使用するアプリケーションに変更することをお薦めしています。ただしリリース 9.0.2 のアプリケーションは、引き続きリリース 9.0.4 で使用できます。

以降では、アプリケーションと mod_osso の統合方法を説明します。SDK の詳細は、付録を参照してください。

2

シングル・サインオン対応のアプリケーションの開発

この章では、mod_osso で動作するアプリケーションの開発方法を説明します。この章の項目は次のとおりです。

- mod_osso を使用したアプリケーションの保護 : 2 つの方法
- mod_osso を使用したアプリケーションの開発
- セキュリティに関する問題 : シングル・サインオフとアプリケーションのログアウト

mod_osso を使用したアプリケーションの保護 : 2 つの方法

mod_osso は、要求された URL が保護されるように構成されている場合に限り、ユーザーを Single Sign-On Server にリダイレクトします。URL は、静的な方法または動的な方法で保護できます。スタティック・ディレクティブは、単純にユーザーの対話に対する制御を mod_osso に渡すことで、アプリケーションを保護します。ダイナミック・ディレクティブは、アプリケーションを保護するだけでなく、アプリケーションにおけるユーザー・アクセスの調整も可能にします。

この項の項目は次のとおりです。

- URL の静的な保護
- ダイナミック・ディレクティブを使用した URL の保護

URL の静的な保護

mod_osso.conf ファイルにディレクティブを適用すると、mod_osso を使用して URL を静的に保護できます。次の例では、Oracle HTTP Server のドキュメント・ルートの真下にある /private ディレクトリを、このディレクティブによって保護する方法を示します。

```
<IfModule mod_osso.c>

  <Location /private>
    AuthType Basic
    require valid-user
  </Location>

</IfModule>
```

エントリを作成したら、ディレクトリにページを追加し、それらをテストします。たとえば、次のように記述します。

```
http://host:port/private/helloworld.html
```

最後に、Oracle HTTP Server を再起動します。

```
$ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
```

ダイナミック・ディレクティブを使用した URL の保護

ダイナミック・ディレクティブは、特殊なエラー・コードを持つ HTTP レスポンス・ヘッダーで、これを使用すると複雑なシングル・サインオン・プロトコルを実装しなくても、アプリケーションではシングル・サインオン・システムの細かい機能を利用できます。mod_osso は、単純な HTTP レスポンスの一部としてアプリケーションからディレクティブを受信すると、適切なシングル・サインオン・プロトコル・メッセージを作成して、Single Sign-On Server に送信します。

OracleAS リリース 9.0.4 でサポートされているダイナミック・ディレクティブは、Java サーブレットと JSP 用です。現在のところ、PL/SQL アプリケーションにはダイナミック・ディレクティブを使用できません。

表 2-1 は、一般的に要求されるダイナミック・ディレクティブの一覧です。

表 2-1 一般的に要求されるダイナミック・ディレクティブ

ディレクティブ	ステータス・コード	ヘッダー
認証リクエスト	401、499	-
強制認証リクエスト	499	`Osso-Paranoid: true`
シングル・サインオフ	470	`Osso-Return-URL` これは、シングル・サインオフの完了後に返す URL です。

mod_osso を使用したアプリケーションの開発

この項では、mod_osso を使用したアプリケーションの作成と有効化の方法について説明します。この項の項目は次のとおりです。

- 静的に保護された PL/SQL アプリケーションの開発
- 静的に保護された Java アプリケーションの開発
- ダイナミック・ディレクティブを使用する Java アプリケーションの開発
- 非 GET 認証について

静的に保護された PL/SQL アプリケーションの開発

以下は、mod_osso で保護された単純なアプリケーションの例です。このアプリケーションは、Single Sign-On Server にユーザーをログインさせ、ユーザー情報を表示し、ユーザーをアプリケーションと Single Sign-On Server の両方からログアウトさせます。

mod_osso を使用して PL/SQL アプリケーションを作成および有効化する手順は、次のとおりです。

1. アプリケーション・プロシージャがロードされるスキーマを作成します。

```
sqlplus sys/sys_password as sysdba
create user schema_name identified by schema_password;
grant connect,  resource to schema_name;
```

2. 次のプロシージャをスキーマにロードし、プロシージャにパブリック・アクセス権を付与します。

```
create or replace procedure show_user_info
 is
 begin
    begin
        htp.init;
     exception
         when others then null;
     end;
     htp.htmlOpen;
     htp.bodyOpen;
     htp.print('<h2>Welcome to Oracle Single Sign-On</h2>');
     htp.print('<pre>');
     htp.print('Remote user: '
        || owa_util.get_cgi_env('REMOTE_USER'));
     htp.print('User DN: '
        || owa_util.get_cgi_env('Osso-User-Dn'));
     htp.print('User Guid: '
        || owa_util.get_cgi_env('Osso-User-Guid'));
     htp.print('Subscriber: '
        || owa_util.get_cgi_env('Osso-Subscriber'));
     htp.print('Subscriber DN: '
        || owa_util.get_cgi_env('Osso-Subscriber-Dn'));
     htp.print('Subscriber Guid: '
        || owa_util.get_cgi_env('Osso-Subscriber-Guid'));
     htp.print('</pre>');
     htp.print('<a href=/osso_logout?'
        ||'p_done_url=http://my.oracle.com>Logout</a>');

     htp.bodyClose;
     htp.htmlClose;
end show_user_info;
```

```
/
show errors;

grant execute on show_user_info to public;
```

3. アプリケーションに対するデータベース・アクセス記述子（DAD）を、$ORACLE_HOME/Apache/modplsql/conf にある dads.conf ファイルに作成します。

```
<Location /pls/DAD_name>
       SetHandler pls_handler
       Order deny,allow
       AllowOverride None
       PlsqlDatabaseConnectString    hostname:port:SID
       PlsqlDatabasePassword         schema_password
       PlsqlDatabaseUsername         schema_name
       PlsqlDefaultPage              schema_name.show_user_info
       PlsqlDocumentTablename        schema_name.wwdoc_document
       PlsqlDocumentPath             docs
       PlsqlDocumentProcedure        schema_name.wwdoc_process.process_
                                       download
       PlsqlAuthenticationMode       Basic
       PlsqlPathAlias                url
       PlsqlPathAliasProcedure       schema_name.wwpth_api_alias.process_
                                       download
       PlsqlSessionCookieName        schema_name
       PlsqlCGIEnvironmentList       OSSO-USER-DN
       PlsqlCGIEnvironmentList       OSSO-USER-GUID
       PlsqlCGIEnvironmentList       OSSO-SUBSCRIBER
       PlsqlCGIEnvironmentList       OSSO-SUBSCRIBER-DN
       PlsqlCGIEnvironmentList       OSSO-SUBSCRIBER-GUID
</Location>
```

4. 次の行を mod_osso.conf ファイルに入力して、アプリケーション DAD を保護します。

```
<Location /pls/DAD_name>
  require valid-user
  authType Basic
</Location>
```

> **注意：** ここでは、mod_osso がシングル・サインオン用にすでに構成されていると想定しています。この手順は、OracleAS のインストール時に実行されます。

5. このアプリケーションで使用する Oracle HTTP Server を再起動します。

```
$ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
```

6. 新しく作成されたファンクションとプロシージャが mod_osso によって保護されているかどうかをテストするには、次のようにブラウザからアクセスします。

   ```
   http://host:port/pls/DAD/schema_name.show_user_info
   ```

   mod_osso.conf が適切に構成されていて、mod_osso が Single Sign-On Server に登録されていれば、URL を選択したときにシングル・サインオン・ログイン・ページが表示されます。

静的に保護された Java アプリケーションの開発

mod_osso を使用してサーブレットや JSP アプリケーションを作成および有効化する手順は、次のとおりです。

1. JSP またはサーブレットを作成します。前述の PL/SQL アプリケーションの例と同様に、次の単純なサーブレットはユーザーのログイン、ユーザー情報の表示およびユーザーのログアウトを行います。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Example servlet showing how to get the SSO User information
 */

public class SSOProtected extends HttpServlet
{

    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");

        // Show authenticated user informationsingle sign-on
        PrintWriter out = response.getWriter();
        out.println("<h2>Welcome to Oracle Single Sign-On</h2>");
        out.println("<pre>");
        out.println("Remote user: "
            + request.getRemoteUser());
        out.println("Osso-User-Dn: "
            +  request.getHeader("Osso-User-Dn"));
        out.println("Osso-User-Guid: "
            +  request.getHeader("Osso-User-Guid"));
        out.println("Osso-Subscriber: "
            +  request.getHeader("Osso-Subscriber"));
```

```
out.println("Osso-User-Dn: "
    +  request.getHeader("Osso-User-Dn"));
out.println("Osso-Subscriber-Dn: "
    +  request.getHeader("Osso-Subscriber-Dn"));
out.println("Osso-Subscriber-Guid: "
    +  request.getHeader("Osso-Subscriber-Guid"));
out.println("Lang/Territory: "
    + request.getHeader("Accept-Language"));
out.println("</pre>");
out.println("<a href=/osso_logout?"
    +"p_done_url=http://my.oracle.com>Logout</a>");
```

2. mod_osso.conf ファイルに次の行を入力して、サーブレットを保護します。

```
<Location /servlet>
   require valid-user
   authType Basic
</Location>
```

3. サーブレットを配置し、Oracle HTTP Server と OC4J を再起動します。

```
$ORACLE_HOME/opmn/bin/opmnctl restartproc type=ohs
$ORACLE_HOME/opmn/bin/opmnctl stopproc type=oc4j
$ORACLE_HOME/opmn/bin/opmnctl startproc type=oc4j
```

4. ブラウザからサーブレットにアクセスを試みて、サーブレットをテストします。URL を選択すると、ログイン・ページが表示されます。

   このプロセスは、次のようになります。まず、ブラウザからサーブレットにアクセスしようとすると、認証を行うために Single Sign-On Server にリダイレクトされます。その後、もう一度サーブレットにリダイレクトされ、ユーザー情報が表示されます。ログアウト・リンクを選択すると、アプリケーションと Single Sign-On Server からログアウトできます。

ダイナミック・ディレクティブを使用する Java アプリケーションの開発

mod_osso による保護は、ダイナミック・ディレクティブとしてアプリケーションに直接書き込まれるため、ダイナミック・ディレクティブを使用するアプリケーションに mod_osso.conf ファイルのエントリは必要ありません。以下のサーブレットは、そのようなディレクティブがどのように組み込まれるかを示しています。静的なアプリケーションと同様に、これらのサンプルの動的なアプリケーションは、ユーザー情報を生成します。

この項の項目は次のとおりです。

- Java の例 1: 簡易認証
- Java の例 2: シングル・サインオフ
- Java の例 3: 強制認証

Java の例 1: 簡易認証

このサーブレットは、request.getRemoteUser() メソッドを使用して、mod_osso Cookie でユーザー名をチェックしています。ユーザー名が存在しない場合、ダイナミック・ディレクティブ 499 という簡易認証リクエストが発行されます。重要な行は、太字で示しています。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Example servlet showing how to use
 * Dynamic Directive for login
 */

public class SSODynLogin extends HttpServlet
{

    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws IOException, ServletException
    {
        String l_user     = null;

        // Try to get the authenticate user name
        try
        {
            l_user = request.getRemoteUser();
        }
        catch(Exception e)
        {
            l_user = null;
        }

        // If user is not authenticated then generate
        // dynamic directive for authentication
        if((l_user == null) || (l_user.length() <= 0) )
        {
           response.sendError(499, "Oracle SSO");
        }
        else
        {
            // Show authenticated user information
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<h2>Welcome to Oracle Single Sign-On</h2>");
            out.println("<pre>");
```

```
            out.println("Remote user: "
                + request.getRemoteUser());
            out.println("Osso-User-Dn: "
                +  request.getHeader("Osso-User-Dn"));
            out.println("Osso-User-Guid: "
                +  request.getHeader("Osso-User-Guid"));
            out.println("Osso-Subscriber: "
                +  request.getHeader("Osso-Subscriber"));
            out.println("Osso-User-Dn: "
                +  request.getHeader("Osso-User-Dn"));
            out.println("Osso-Subscriber-Dn: "
                +  request.getHeader("Osso-Subscriber-Dn"));
            out.println("Osso-Subscriber-Guid: "
                +  request.getHeader("Osso-Subscriber-Guid"));
            out.println("Lang/Territory: "
                + request.getHeader("Accept-Language"));
            out.println("</pre>");
        }
    }
```

注意： Oracle JAAS Provider を使用している場合、499 のかわりにディレクティブ・コード 401 を使用できます。

Java の例 2: シングル・サインオフ

このサーブレットは、ユーザーがアプリケーションにあるログイン・リンクを選択した際に起動します。アプリケーションはサインオフ完了後に戻る URL を設定します。そして、ユーザーをシングル・サインオフ・ページへ移動させるディレクティブを発行します。重要な行は、太字で示しています。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Example servlet showing how to use
 * Dynamic Directive for logout
 */

public class SSODynLogout extends HttpServlet
{
    public void service (HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException
    {
        // Set the return URL
```

```
        response.setHeader("Osso-Return-Url",
               "http://my.oracle.com" );
        // Send Dynamic Directive for logout
        response.sendError(470, "Oracle SSO");
    }
}
```

注意： 別の方法として、同じコンピュータの osso_logout URL にリダイレクトすることもできます。

Java の例 3: 強制認証

ログインしているユーザーがタイムアウトになった場合に、アプリケーションでそのユーザーを強制的に再認証させることができます。再認証のディレクティブは、次のサーブレットに記述されています。重要な行は、太字で示しています。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Example servlet showing how to use
 * Dynamic Directive for forced login
 */

public class SSODynForcedLogin extends HttpServlet
{

    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws IOException, ServletException
    {
        String l_user = null;
        // Try to get the authenticate user name
        try
        {
            l_user = request.getRemoteUser();
        }
        catch(Exception e)
        {
            l_user = null;
        }

        // If the user is authenticated then show
        // user information; otherwise generate Dynamic
        // Directive for forced login
```

```
            if(l_user != null)
            {
                // Show authenticated user information
                PrintWriter out = response.getWriter();
                response.setContentType("text/html");
                out.println("<h2>Welcome to Oracle Single Sign-On.</h2>");
                out.println("<pre>");
                out.println("Remote user: "
                    + request.getRemoteUser());
                out.println("Osso-User-Dn: "
                    +  request.getHeader("Osso-User-Dn"));
                out.println("Osso-User-Guid: "
                    +  request.getHeader("Osso-User-Guid"));
                out.println("Osso-Subscriber: "
                    +  request.getHeader("Osso-Subscriber"));
                out.println("Osso-User-Dn: "
                    +  request.getHeader("Osso-User-Dn"));
                out.println("Osso-Subscriber-Dn: "
                    +  request.getHeader("Osso-Subscriber-Dn"));
                out.println("Osso-Subscriber-Guid: "
                    +  request.getHeader("Osso-Subscriber-Guid"));
                out.println("Lang/Territory: "
                    + request.getHeader("Accept-Language"));
                out.println("</pre>");
            }
            else
            {
                response.setHeader( "Osso-Paranoid", "true" );
                response.sendError(499, "Oracle SSO");
            }
        }
    }
```

非 GET 認証について

mod_osso で保護されたアプリケーションの 1 ページ目は、GET 認証方式を使用する URL であることが必要です。POST メソッドを使用すると、Single Sign-On Server にリダイレクトしている間に、ユーザーがログイン時に入力したデータが失われます。グローバル・ユーザーの非アクティブ・タイムアウトを有効にするかどうかを決定する際は、ユーザーのリダイレクトが、タイムアウト後に再びログインしてから行われることに注意してください。

セキュリティに関する問題 : シングル・サインオフとアプリケーションのログアウト

OracleAS リリース 9.0.4 を使用してカスタム・アプリケーションを構築する場合は、グローバル・ログアウトまたはシングル・サインオフの実行時に、シングル・サインオン Cookie と mod_osso Cookie のみがクリアされることに注意してください。つまり、OracleAS アプリケーションをコーディングする際は、シングル・サインオン・ユーザー名とレルム名が OC4J セッションまたはアプリケーション・セッションに保存されるようにする必要があります。アプリケーションではその後、これらの値と、mod_osso から渡された値とを比較し、値が一致した場合は、パーソナライズされたコンテンツを表示する必要があります。値が一致しない場合、すなわち mod_osso Cookie が存在しない場合は、アプリケーション・セッションをクリアし、ユーザーにログインを強制する必要があります。

この項の項目は次のとおりです。

- アプリケーションのログイン : コード例
- アプリケーションのログアウト : 推奨コード

アプリケーションのログイン : コード例

最初の 2 つのコード例には、前述の項で説明したロジックが使用されていません。このロジックは、3 つ目の例で使用されています。これらは Java の例ですが、Perl、PL/SQL、CGI などの言語で記述することもできます。

不適切なコード例 1

```
// Get user name from application session. This session was
// established by the application cookie or OC4J session cookie
String username = request.getSession().getAttribute('USER_NAME');

// Get subscriber name from application session. This session was
// established by the application cookie or OC4J session cookie.
String subscriber = request.getSession().getAttribute('SUBSCRIBER_NAME');

// Get user security information from application session. This session was
established  by the application cookie or OC4J session cookie
String user_sec_info = request.getSession().getAttribute('USER_APP_SEC');

if((username != null) && (subscriber!= null))
{
  // Show personalized user content
  show_personalized_page(username, subscriber, user_sec_info);
}
else
{
  // Send Dynamic Directive for login
```

```
  response.sendError( 499, "Oracle SSO" );
```

不適切なコード例 2

```
// Get SSO username from http header
String username = request.getRemoteUser();

// Get subscriber name from SSO http header
String subscriber = request.getHeader('OSSO-SUBSCRIBER');

// Get user security information from application session.  This session
// was established by the application or OC4J session
String user_sec_info =request.getSession().getAttribute('USER_APP_SEC');

if((ssousername != null)&&(subscriber!= null))
{
  // Show personalized user content
show_personalized_page(username, subscriber, user_sec_info);
}
else
{
  // Send Dynamic Directive for login
response.sendError( 499, "Oracle SSO" );
}
```

推奨コード

```
// Get user name from application session. This session was
// established by the application or OC4J session
String username = request.getSession().getAttribute('USER_NAME');

// Get subscriber name from application session. This session was
// established by the application or OC4J session
String subscriber = request.getSession().getAttribute('SUBSCRIBER_NAME');

// Get user security information from application session. This session
// was established by the application or OC4J session
String user_sec_info = request.getSession().getAttribute('USER_APP_SEC');

// Get username and subscriber name from JAZN API */
JAZNUserAdaptor jaznuser  = (JAZNUserAdaptor)requset.getUserPrincipal();
     String ssousername   = jaznuser.getName();
     String ssosubscriber = jaznuser.getRealm().getName();

// If you are not using JAZN api then you can also get the username and
// subscriber name from mod_osso headers
String ssousername   = request.getRemoteUser();
```

```
String ssosubscriber = request.getHeader('OSSO-SUBSCRIBER');

// Check for application session. Create it if necessary.
if((username == null) || (subscriber == null) )
  {
    ...Code to create application session. Get the user information from
    JAZN api(or mod_osso headers if you are not using JAZN api) and
    populate the application session with user, subscriber and user
    security info...
  }

if((username != null)&&(subscriber != null)
  &&(ssousername != null)&&(ssosubscriber != null)
  &&(username.equalsIgnoreCase(ssousername) == 0 )
  &&(subscriber.equalsIgnoreCase(ssosubscriber) == 0))
{
  // Show personalized user content
show_personalized_page(username, subscriber, user_sec_info);
}
else
{
     ...Code to Wipe-out application session, followed by...

// Send Dynamic Directive for login
// If you are using JAZN then you should use following code
// response.sendError( 401);

// If you are not using JAZN api then you should use following code
// response.sendError( 499, "Oracle SSO" );
}
```

アプリケーションのログアウト : 推奨コード

ユーザーを認証するアプリケーションのほとんどには、ログアウト・リンクがあります。シングル・サインオン対応のアプリケーションでは、ログアウト・ハンドラ内の他のコードに加えて、ログアウト用のダイナミック・ディレクティブが起動します。ユーザーがログアウト・ディレクティブを起動すると、シングル・サインオフまたはグローバル・ログアウトが実行されます。次の例では、Java でシングル・サインオフを記述した場合のコードを示しています。

```
// Clear application session, if any
String l_return_url := return url to your application e.g. home page
response.setHeader( "Osso-Return-Url", l_return_url);
response.sendError( 470, "Oracle SSO" );
```

A

Single Sign-On Software Development Kit

Single Sign-On SDK は、PL/SQL API と Java API で構成されます。パートナ・アプリケーションを作成するには、これらの API を使用します。API の実装方法を示すコードは、付録 B「PL/SQL API および Java API の使用方法」を参照してください。

この付録の項目は次のとおりです。

- PL/SQL API
- Java API

> **注意：** SDK は推奨されていません。リリース 9.0.2 の SDK を使用してアプリケーションを構築した場合、オラクル社では、mod_osso のアプリケーションに変更することをお薦めしています。ただしリリース 9.0.2 のアプリケーションは、引き続きリリース 9.0.4 で使用できます。

PL/SQL API

この項の項目は次のとおりです。

- ファンクションとプロシージャ
- 表の定義
- 例外

ファンクションとプロシージャ

この項のファンクションとプロシージャは、WWSEC_SSO_ENABLER パッケージの一部です。PL/SQL アプリケーションをパートナ・アプリケーションとして使用するときは、このパッケージを利用します。

この項では、次のファンクションおよびプロシージャを扱います。

- GENERATE_REDIRECT ファンクション
- PARSE_URL_COOKIE プロシージャ
- GET_ENABLER_CONFIG プロシージャ
- CREATE_ENABLER_CONFIG プロシージャ
- UPDATE_ENABLER_CONFIG プロシージャ
- DELETE_ENABLER_CONFIG プロシージャ
- ENCRYPT_COOKIE ファンクション
- DECRYPT_COOKIE ファンクション

GENERATE_REDIRECT ファンクション

このファンクションは、サーバーによって解析される SITE2PSTORETOKEN とともに、リダイレクト URL を生成します。

構文

```
FUNCTION GENERATE_REDIRECT
(
     P_LSNR_TOKEN    IN  VARCHAR2
   , P_URL_REQUESTED IN  VARCHAR2
   , P_URL_CANCEL    IN  VARCHAR2
   , P_FORCED_AUTH   IN  NUMBER DEFAULT SIMPLE_AUTH
) RETURN VARCHAR2;
```

表 A-1 GENERATE_REDIRECT のパラメータ

パラメータ	説明
P_LSNR_TOKEN	パートナ・アプリケーションの登録情報を取得するリスナー・トークン。リスナー・トークンは、現在のリクエストの URL に使用されるホスト名とポートで構成されます。このトークンは、WWSEC_ENABLER_CONFIG_INFO$ 表から適切な構成エントリを選択するときに使用されます。
P_URL_REQUESTED	クライアントが要求した URL。 URL パラメータが含まれる場合は、エンコードされた URL であることが必要です。たとえば、次のように記述します。 http://*host*:*port*/jsp/order.jsp?itemid=1234&type=purchase
P_URL_CANCEL	ユーザーがログイン・ページで「取消」をクリックした場合にリダイレクトされる URL。 URL パラメータが含まれる場合は、エンコードされた URL であることが必要です。たとえば、次のように記述します。 http://*host*:*port*/jsp/order.jsp?itemid=1234&type=purchase
P_FORCED_AUTH	強制認証フラグ。
REDIRECTURL	パートナ・アプリケーションでブラウザをリダイレクトする URL。この URL には、認証リクエストが含まれます。

例

```
WWSEC_SSO_ENABLER.GENERATE_REDIRECT
(
  p_lsnr_token    => listener token
  p_url_requested => requested url
  p_url_cancel    => cancel url
  p_forced_auth   => forced authentication flag
  redirecturl     => redirect url
);
```

PARSE_URL_COOKIE プロシージャ

このプロシージャは、サーバー側の GENERATE_REDIRECT ファンクションによって生成された URL Cookie を解析します。

構文

```
PROCEDURE parse_url_cookie
(
     P_LSNR_TOKEN            IN   VARCHAR2
   , P_ENC_URL_COOKIE        IN   VARCHAR2
   , P_URL_REQUESTED         OUT  VARCHAR2
   , P_SSO_USERNAME          OUT  VARCHAR2
   , P_SSO_USER_DN           OUT  VARCHAR2
   , P_SSO_USER_GUID         OUT  VARCHAR2
   , P_SUBSCRIBER_NAME       OUT  VARCHAR2
   , P_SUBSCRIBER_DN         OUT  VARCHAR2
   , P_SUBSCRIBER_GUID       OUT  VARCHAR2
   , P_USER_IPADDRESS        OUT  VARCHAR2
   , P_SSO_TIMEREMAINING     OUT  NUMBER
   , P_NLS_LANGUAGE          OUT  VARCHAR2
   , P_NLS_TERRITORY         OUT  VARCHAR2
);
```

表 A-2 PARSE_URL_COOKIE のパラメータ

パラメータ	説明
P_LSNR_TOKEN	リスナー・トークン。
P_ENC_URL_COOKIE	暗号化された URL Cookie。
P_URL_REQUESTED	要求された URL。
P_SSO_USERNAME	認証されたユーザー名。
P_SSO_USER_DN	認証されたユーザー DN。
P_SSO_USER_GUID	認証されたユーザー GUID。
P_SUBSCRIBER_NAME	サブスクライバ名。
P_SUBSCRIBER_DN	サブスクライバ DN。
P_SUBSCRIBER_GUID	サブスクライバ GUID。
P_USER_IPADDRESS	ユーザー・マシンの IP アドレス。
P_SSO_TIMEREMAINING	残りのセッション期間。
P_NLS_LANGUAGE	ユーザーの言語の選択。
P_NLS_TERRITORY	ユーザーの地域の選択。

例

```
WWSEC_SSO_ENABLER.PARSE_URL_COOKIE
(
   p_lsnr_token        => listener token
   p_enc_url_cookie    => encrypted URL cookie
   p_url_requested     => requested URL
   p_sso_username      => authenticated SSO username
   p_sso_user_dn       => authenticated SSO user DN
   p_sso_user_guid     => authenticated SSO user GUID
   p_subscriber_name   => subscriber name
   p_subscriber_dn     => subscriber DN
   p_subscriber_guid   => subscriber GUID
   p_user_ipaddress    => ipaddress of the sso user's machine
   p_sso_timeremaining => remaining single sign-on session duration
   p_nls_language      => language selection of sso user
   p_nls_territory     => territory selection of sso user
 );
```

GET_ENABLER_CONFIG プロシージャ

このプロシージャは、リスナー・トークンに指定されているパートナ・アプリケーションの登録情報を返します。

構文

```
PROCEDURE GET_ENABLER_CONFIG
(
    P_LSNR_TOKEN            IN   VARCHAR2,
    P_SITE_TOKEN            OUT  VARCHAR2,
    P_SITE_ID               OUT  VARCHAR2,
    P_LS_LOGIN_URL          OUT  VARCHAR2,
    P_LS_LOGOUT_URL         OUT  VARCHAR2,
    P_URL_COOKIE_VERSION    OUT  VARCHAR2,
    P_ENCRYPTION_KEY        OUT  VARCHAR2,
    P_IPADDR_CHECK          OUT  VARCHAR2
);
```

表 A-3 GET_ENABLER_CONFIG のパラメータ

パラメータ	説明
P_LSNR_TOKEN	リスナー・トークン。
P_SITE_TOKEN	サイト・トークン。
P_SITE_ID	サイト・トークン。
P_LS_LOGIN_URL	ログイン URL。

表 A-3 GET_ENABLER_CONFIG のパラメータ（続き）

パラメータ	説明
P_LS_LOGOUT_URL	シングル・サインオフ URL。
P_URL_COOKIE_VERSION	URL Cookie のバージョン。
P_ENCRYPTION_KEY	暗号化鍵。
P_IPADDR_CHECK	IP アドレスを検証するかどうかを指定します。

例

```
WWSEC_SSO_ENABLER_PRIVATE.GET_ENABLER_CONFIG
(
   p_lsnr_token          => listener token
   p_site_token          => site token
   p_site_id             => site token
   p_ls_login_url        => login url of SSO Server
   p_ls_logout_url       => Single Sign-Off URL of SSO Server
   p_url_cookie_version  => url cookie version
   p_encryption_key      => encryption key
   p_ipaddr_check        => if ip address should be verified
```

CREATE_ENABLER_CONFIG プロシージャ

このプロシージャは、リスナー・トークンに指定されているパートナ・アプリケーション登録情報を、イネーブラ構成表に保存します。

構文

```
PROCEDURE CREATE_ENABLER_CONFIG
(
    P_LSNR_TOKEN           IN  VARCHAR2,
    P_SITE_TOKEN           IN  VARCHAR2,
    P_SITE_ID              IN  VARCHAR2,
    P_LS_LOGIN_URL         IN  VARCHAR2,
    P_LS_LOGOUT_URL        IN  VARCHAR2,
    P_URL_COOKIE_VERSION   IN  VARCHAR2,
    P_ENCRYPTION_KEY       IN  VARCHAR2,
    P_IPADDR_CHECK         IN  VARCHAR2
);
```

表 A-4 CREATE_ENABLER_CONFIG のパラメータ

パラメータ	説明
P_LSNR_TOKEN	リスナー・トークン。
P_SITE_TOKEN	サイト・トークン。
P_SITE_ID	サイト・トークン。
P_LS_LOGIN_URL	ログイン URL。
P_LS_LOGOUT_URL	シングル・サインオフ URL。
P_URL_COOKIE_VERSION	URL Cookie のバージョン。
P_ENCRYPTION_KEY	暗号化鍵。
P_IPADDR_CHECK	IP アドレスを検証するかどうかを指定します。

例

```
WWSEC_SSO_ENABLER.CREATE_ENABLER_CONFIG
(
   p_lsnr_token          =>  listener token
   p_site_token          =>  site token
   p_site_id             =>  site token
   p_ls_login_url        =>  login url of SSO Server
   p_ls_logout_url       =>  Single Sign-Off URL of the single sign-on server
   p_url_cookie_version  =>  URL cookie version
   p_encryption_key      =>  Encryption key
   p_ipaddr_check        =>  If IP address should be verified
)
```

UPDATE_ENABLER_CONFIG プロシージャ

このプロシージャは、リスナー・トークンに指定されているパートナ・アプリケーションの登録情報を変更します。

構文

```
PROCEDURE UPDATE_ENABLER_CONFIG
(
    P_LSNR_TOKEN           IN  VARCHAR2,
    P_SITE_TOKEN           IN  VARCHAR2,
    P_SITE_ID              IN  VARCHAR2,
    P_LS_LOGIN_URL         IN  VARCHAR2,
    P_LS_LOGOUT_URL        IN  VARCHAR2,
    P_URL_COOKIE_VERSION   IN  VARCHAR2,
    P_ENCRYPTION_KEY       IN  VARCHAR2,
    P_IPADDR_CHECK         IN  VARCHAR2
);
```

表 A-5 UPDATE_ENABLER_CONFIG のパラメータ

パラメータ	説明
P_LSNR_TOKEN	リスナー・トークン。
P_SITE_TOKEN	サイト・トークン。
P_SITE_ID	サイト・トークン。
P_LS_LOGIN_URL	ログイン URL。
P_LS_LOGOUT_URL	シングル・サインオフ URL。
P_URL_COOKIE_VERSION	URL Cookie のバージョン。
P_ENCRYPTION_KEY	暗号化鍵。
P_IPADDR_CHECK	IP アドレスを検証するかどうかを指定します。

例

```
WWSEC_SSO_ENABLER.UPDATE_ENABLER_CONFIG
(
   p_lsnr_token          =>  listener token
   p_site_token          =>  site token
   p_site_id             =>  site token
   p_ls_login_url        =>  login url of SSO Server
   p_ls_logout_url       =>  Single Sign-Off URL of SSO Server
   p_url_cookie_version  =>  url cookie version
   p_encryption_key      =>  encryption key
   p_ipaddr_check        =>  if IP address should be verified or not
)
```

DELETE_ENABLER_CONFIG プロシージャ

このプロシージャは、リスナー・トークンに指定されているパートナ・アプリケーションの登録情報を削除します。

構文

```
PROCEDURE DELETE_ENABLER_CONFIG
(
    P_LSNR_TOKEN  IN VARCHAR2
);
```

表 A-6 DELETE_ENABLER_CONFIG のパラメータ

パラメータ	説明
P_LSNR_TOKEN	リスナー・トークン。パートナ・アプリケーションの登録情報を取得します。

例

```
WWSEC_SSO_ENABLER.DELETE_ENABLER_CONFIG
(
   p_lsnr_token  =>  listener token
);
```

ENCRYPT_COOKIE ファンクション

このファンクションは、暗号化 Cookie 本体を返します。

構文

```
FUNCTION ENCRYPT_COOKIE
(
    p_lsnr_token  in  varchar2,
    p_cookie      in  varchar2
 )  return varchar2;
```

表 A-7 ENCRYPT_COOKIE のパラメータ

パラメータ	説明
P_LSNR_TOKEN	リスナー・トークン。パートナ・アプリケーションの登録情報を取得します。

例

```
WWSEC_SSO_ENABLER.ENCRYPT_COOKIE
(
  p_lsnr_token  =>  listener token
  p_enc_cookie  =>  cookie value to be encrypted
)
```

DECRYPT_COOKIE ファンクション

このファンクションは、暗号化 Cookie から復号化 Cookie を返します。

構文

```
(
    P_LSNR_TOKEN  IN  VARCHAR2,
    P_ENC_COOKIE  IN  VARCHAR2
 )  RETURN VARCHAR2;
```

表 A-8 DECRYPT_COOKIE のパラメータ

パラメータ	説明
P_LSNR_TOKEN	リスナー・トークン。パートナ・アプリケーションの登録情報を取得します。
P_ENC_COOKIE	暗号化される Cookie の値。

例

```
WWSEC_SSO_ENABLER.DECRYPT_COOKIE
(
   p_lsnr_token  =>  listener token
   p_enc_cookie  =>  cookie value to be encrypted
)
```

表の定義

Single Sign-On SDK には、WWSEC_ENABLER_CONFIG_INFO$ と WWSEC_SSO_LOG$ という、2 つのパートナ・アプリケーション用の表があります。1 つ目の表は、アプリケーションが接続先の Single Sign-On Server を判定できるようにするための構成情報を格納しています。もう一方の表は、クライアント側でのデバッグ情報を格納しています。これはデバッグを有効にするとアクセスできます。

WWSEC_ENABLER_CONFIG_INFO$

```
CREATE TABLE wwsec_enabler_config_info$
(
  lsnr_token              VARCHAR2(255)
  , site_token            VARCHAR2(255)
  , site_id               VARCHAR2(255)
  , ls_login_url          VARCHAR2(1000)
  , urlcookie_version     VARCHAR2(80)
  , encryption_key        VARCHAR2(1000)
  , encryption_mask_pre   VARCHAR2(1000)
  , encryption_mask_post  VARCHAR2(1000)
  , url_cookie_ip_check   VARCHAR2(1)
  );
```

WWSEC_SSO_LOG$

```
CREATE TABLE wwsec_sso_log$
(
  , SUBSCRIBER_ID  NUMBER  NOT NULL
  , id           NUMBER
  , msg          VARCHAR2(1000)
  , log_date     DATE
 );
```

例外

表 A-9 は、PL/SQL のファンクションとプロシージャで発生する例外とその説明の一覧です。

表 A-9 例外

例外	説明
UNKNOWN_ERROR_EXCEPTION	一般的なエラー。
CONFIG_MISSING_EXCEPTION	SDK の構成テーブルが入力されていないか、テーブルのコンテンツが無効です。
DUP_CONFIG_EXCEPTION	同じリスナー・トークンを持ったパートナ構成がすでに存在しています。
ENCRYPTION_FAILED_EXCEPTION	不適切な鍵または無効な入力データ。
DECRYPTION_FAILED_EXCEPTION	不適切な鍵または無効な入力データ。
UNSUPPORTED_VERSION_EXCEPTION	SDK のバージョンと Single Sign-On Server のバージョンとの間に互換性がありません。
IPADDR_ERROR_EXCEPTION	認証前と認証後のアドレスが一致しません。ユーザーがプロキシ経由でアプリケーションにアクセスしている、セキュリティ上の攻撃を受けている、あるいはユーザーのコンピュータが固定 IP アドレスを使用していない可能性があります。
COOKIE_EXPIRED_EXCEPTION	Single Sign-On Server が送信した認証トークンがタイムアウトになりました。
NULL_ATTRIBUTE_EXCEPTION	無効な入力データ。

Java API

Java API を PL/SQL API のかわりに使用して、パートナ・アプリケーションを作成することができます。Java API の使用方法は、『Oracle Application Server Single Sign-On API Reference』を参照してください。

B

PL/SQL API および Java API の使用方法

この付録では、シングル・サインオン対応のパートナ・アプリケーションを有効化する方法を示す、サンプル・プログラムを紹介します。

この付録の項目は次のとおりです。

- API を使用する前に
- PL/SQL API を使用したパートナ・アプリケーションの作成
- Java API を使用したパートナ・アプリケーションの作成

API を使用する前に

Single Sign-On SDK を使用してパートナ・アプリケーションを作成する前に、Single Sign-On SDK が正しくインストールされ、構成されていることを確認します。付属のインストールと構成の手順に従います。SDK は $ORACLE_HOME/sso/lib/ssosdk902.zip に含まれています。

PL/SQL API を使用したパートナ・アプリケーションの作成

以下の例は、PL/SQL API を使用したパートナ・アプリケーションの開発方法を示しています。データベース・アクセス記述子の作成方法は、『Oracle Application Server 10*g* mod_plsql ユーザーズ・ガイド』を参照してください。PL/SQL アプリケーションの作成方法は、『Oracle Application Server 10*g* PL/SQL Web Toolkit リファレンス』を参照してください。例には、SAMPLE_SSO_PAPP.SSOAPP、SAMPLE_SSO_PAPP.SIGN_ON、SAMPLE_SSO_PAPP.LOGOUT の 3 つのプロシージャがあります。

SAMPLE_SSO_PAPP.SSOAPP

このプロシージャでは、アプリケーションの URL を構築します。このプロシージャでは、アプリケーション Cookie が存在し、ユーザー情報が取得できるかどうかをチェックします。アプリケーション Cookie が存在しない場合は、リダイレクト URL を生成して、ユーザーを Single Sign-On Server にリダイレクトします。

SAMPLE_SSO_PAPP.SIGN_ON

このプロシージャは、Single Sign-On Server から URLC トークンを取得して、そのトークンを解読し、ユーザー情報と要求された URL を取得します。また、アプリケーション Cookie を設定し、ブラウザをパートナ・アプリケーションの URL にリダイレクトします。

SAMPLE_SSO_PAPP.LOGOUT

このプロシージャは、アプリケーションのログアウト URL を実装します。

パッケージ papp.pks および papp.pkb のサンプル・コードは、demo/plsql 内のファイル ssosdk902.zip に含まれています。

注意： リクエスト URL および取消 URL は、その中に URL パラメータが含まれている場合、エンコードされた URL であることが必要です。たとえば、次のように記述します。

```
http://host:port/dad/schema.procedure?itemid=1234&type=purchase
```

PL/SQL では、URL のエンコードに `wwutl_htf.encode` プロシージャを使用できます。

Java API を使用したパートナ・アプリケーションの作成

パートナ・アプリケーションはまず、ユーザーを Single Sign-On Server にリダイレクトして、認証を要求します。認証に成功すると、そのアプリケーション・セッション Cookie を設定します。2 回目以降は、そのアプリケーション・セッション Cookie を検証します。アプリケーション・セッション Cookie が見つからない場合、パートナ・アプリケーションはユーザーを Single Sign-On Server にリダイレクトします。サーバーでユーザー・リクエストを毎回検証する必要がないようにするには、すべてのパートナ・アプリケーションで自身のアプリケーション・セッションを保持する必要があります。

この項では、サーブレットおよび JavaServer Pages（JSP）で使用できる汎用 Bean の実装方法を示します。この項の項目は次のとおりです。

- サーブレットのパートナ・アプリケーション
- JSP のパートナ・アプリケーション

注意： リクエスト URL および取消 URL は、その中に URL パラメータが含まれている場合、エンコードされた URL であることが必要です。たとえば、次のように記述します。

```
http://host:port/jsp/order.jsp?itemid=1234&type=purchase
```

Java では、URL のエンコードに java.net.URLEncoder クラスを使用できます。

サーブレットのパートナ・アプリケーション

ここで示す Java サーブレットの例は、ssosdk902.zip に含まれるファイルで構成されます。これらのファイルは次のとおりです。

- Bean クラス

  demo/java/beans 内のファイル SSOEnablerBean.java および SSOEnablerServletBean.java。配置に合せて、これらのファイルを編集します。

- サーブレット・クラス

  demo/java/servlet 内のファイル SSOPartnerServlet.java、SSOSignOnServlet.java および SSOPartnerLogoutServlet.java。

アプリケーションにアクセスするには、これらの Bean ファイルとサーブレット・ファイルをコンパイルして、OC4J に配置する必要があります。

このアプリケーションの認証フローは次のとおりです。

1. ユーザーは、SSOPartnerServlet アプリケーションの URL を開きます。このサーブレットは、SSOEnablerServletBean を利用してユーザー情報を取得します。ユーザー情報を取得できた場合、アプリケーション内で使用し、取得できなかった場合、ブラウザはユーザーを Single Sign-On Server にリダイレクトします。
2. Single Sign-On Server は、認証に成功すると、次の処理を実行します。
   - ユーザーを SSOSignOnServlet URL にリダイレクトして、アプリケーション Cookie を設定します。
   - SSOEnablerServletBean を使用して、要求されたアプリケーション URL にユーザーをリダイレクトします。サーブレットは、アプリケーション Cookie を使用して、ユーザー情報を表示します。

JSP のパートナ・アプリケーション

JSP パートナ・アプリケーションの例も、ssosdk902.zip に含まれるファイルで構成されます。これらのファイルは次のとおりです。

- Bean クラス

  demo/java/beans 内のファイル SSOEnablerJspBean.java および SSOEnablerServletBean.java。配置に合せて、これらのファイルを編集します。

- JSP ファイル

  demo/java/jsp 内のファイル ssoinclude.jsp、ssosignon.jsp、papp.jsp および papplogoff.jsp。

アプリケーションにアクセスするには、これらの Bean Java ファイルをコンパイルして、JSP ファイルとともに OC4J に配置する必要があります。コンパイルの詳細は、ssosdk902.zip を参照してください。

このアプリケーションの認証フローは次のとおりです。

1. ユーザーが papp.jsp ページを開きます。
2. papp.jsp は、ssoinclude.jsp ページを利用して、ユーザー情報を取得します。ユーザー情報を取得できた場合、アプリケーションで使用し、取得できなかった場合、SSOEnablerJspBean を使用して、ユーザーを Single Sign-On Server にリダイレクトします。
3. Single Sign-On Server は、認証に成功したら、ユーザーを ssosignon.jsp ページにリダイレクトします。このページは、アプリケーション Cookie を設定し、SSOEnablerJspBean を使用して、要求されたアプリケーション URL にユーザーをリダイレクトします。

C

SDK を使用したアプリケーションの追加と編集

Single Sign-On SDK に統合されたアプリケーションの登録および編集には、「SSO Server 管理」ページのリンクとしてアクセス可能な「パートナ・アプリケーションの管理」ページを使用します。このページでは、mod_osso 構成情報も表示できます。リリース 9.0.2 以上では、mod_osso がインストーラによって自動的に登録されます。

SDK を使用したパートナ・アプリケーションを、UI を使用して手動で登録する場合、アプリケーションは Single Sign-On Server にのみ登録されることに注意してください。アプリケーションをパートナ・アプリケーションのデータベースに登録するには、シングル・サインオン UI から登録情報を手動でコピーする必要があります。

この付録の項目は次のとおりです。

- 「パートナ・アプリケーションの追加」ページ
- 「パートナ・アプリケーションの編集」ページ

「パートナ・アプリケーションの追加」ページ

管理ページで「パートナ・アプリケーションの追加」リンクを選択すると、「パートナ・アプリケーションの作成」ページが表示されます。このページのフィールドを使用して、Single Sign-On Server にアプリケーションを登録します。各フィールドの説明は、次の表を参照してください。追加したアプリケーションは、すでに保存されているアプリケーションとともに日付別に表示されます。

表 C-1 パートナ・アプリケーションへのログイン

フィールド	説明
名前	パートナ・アプリケーションの一意な名前を入力します。
ホーム URL	アプリケーションのホームページの URL を入力します。
成功 URL	パートナ・アプリケーションのセッションおよびセッション Cookie を設定するルーチンの URL を入力します。このルーチンは、ユーザーが最初に要求した URL にブラウザをリダイレクトします。 この URL は、Single Sign-On Server のユーザー識別情報を処理するプロシージャを指す必要があります。また、URL には接頭辞 `http://` を付けます。次の例は、OracleAS Portal の成功時の URL です。 `http://server.domain.com:5000/pls/DAD/portal.wwsec_app_priv.process_signon`
ログアウト URL	アプリケーションのログアウト・ルーチンの URL を入力します。この URL は、他のページとともにシングル・サインオフ・ページから表示されるため、ユーザーはサーバーとアクティブなパートナ・アプリケーションから同時にログアウトできます。

表 C-2 ログイン可能期間

フィールド	説明
開始日	ユーザーが Single Sign-On Server を使用してパートナ・アプリケーションにアクセスできる最初の日付を入力します。日付の書式はフィールド・ラベルの横に表示されたものを使用します。
終了日	ユーザーが Single Sign-On Server を使用してパートナ・アプリケーションにアクセスできる最後の日付を入力します。日付の書式はフィールド・ラベルの横に表示されたものを使用します。 **注意 :** このフィールドを空白にすると、パートナ・アプリケーションを無期限に使用できます。

表 C-3 アプリケーション管理者

フィールド	説明
管理者の電子メール	パートナ・アプリケーション管理者の電子メール・アドレスを入力します。
管理者の情報	パートナ・アプリケーション管理者に関して、必要な追加情報を入力します。

「パートナ・アプリケーションの編集」ページ

「パートナ・アプリケーションの編集」ページには、「パートナ・アプリケーションの作成」ページに表示されるすべてのフィールドが含まれ、「パートナ・アプリケーションへのログイン」セクションにさらに 5 つのフィールドが追加されています。これら 5 つのフィールドについては、C-3 ページの表 C-4 を参照してください。

表 C-4「パートナ・アプリケーションの編集」ページのフィールド

フィールド	説明
ID	「ID」の値は、パートナ・アプリケーションが追加されると自動的に設定されます。これは、Single Sign-On Server によるパートナ・アプリケーションの識別に使用されます。
トークン	トークンは、パートナ・アプリケーションが追加されると自動的に設定されます。これは、Single Sign-On Server によるパートナ・アプリケーションの識別に使用されます。パートナ・アプリケーションは、認証を要求するときにアプリケーション・トークンを使用して、自身を Single Sign-On Server に認識させる必要があります。
暗号化キー	パートナ・アプリケーションの暗号化鍵。
ログイン URL	「成功 URL」と同じで、アプリケーションのセッションと Cookie を設定します。
シングル・サインオフ URL	アプリケーションのログアウトに使用する URL と同じです。

パートナ・アプリケーションを編集する手順は、次のとおりです。

1. 「パートナ・アプリケーションの管理」ページで、「パートナ・アプリケーションの編集 / 削除」ヘッダーの下の一覧からアプリケーションを選択します。
2. そのアプリケーションの「編集」リンクをクリックします。このリンクは鉛筆型のアイコンで表示されます。
3. 「パートナ・アプリケーションの編集」ページで、表 C-1 に示すフィールドの値を編集します。このページで編集できる値はこれらのフィールド値のみです。

4. 現行画面の変更を保存し、画面を更新するには、「適用」をクリックします。すべての変更を保存し、前の画面に戻るには、「OK」をクリックします。

D

パートナ・アプリケーションに渡されるユーザー属性

表 D-1 は、mod_osso からアプリケーションに渡されるすべてのユーザー属性の一覧です。この表では、鍵やハンドルとして使用したり、Oracle Internet Directory から他のユーザー属性を取得したりする場合に、それぞれの属性が推奨されるかどうかも示しています。

表 D-1 パートナ・アプリケーションに渡されるユーザー属性

HTTP ヘッダー名	説明	ソース	鍵またはハンドルとしての使用
Osso-User-Guid	シングル・サインオン・ユーザーのグローバルに一意なユーザー ID（GUID）	シングル・サインオン・ユーザーのグローバルに一意なユーザー ID（GUID）	推奨。
Osso-Subscriber-Guid	レルムの GUID	Oracle Internet Directory のレルム・エントリ	推奨。
Remote-User	ログイン・ページでユーザーが入力したユーザー・ニックネーム	シングル・サインオン・ログイン・ページ	9.0.3 以下のアプリケーションには推奨（9.0.4 アプリケーションには非推奨）。
Osso-Subscriber	レルムのわかりやすい名前	Oracle Internet Directory のレルム・エントリ	9.0.3 以下のアプリケーションには推奨（9.0.4 アプリケーションには非推奨）。
Accept-Language	ISO 形式の言語と地域	Single Sign-On Server	非適用。
Osso-User-Dn	シングル・サインオン・ユーザーの識別名（DN）	Oracle Internet Directory のユーザー・エントリ	非推奨。Oracle Internet Directory でのユーザー検索には GUID ヘッダーを使用してください。[1]
Osso-Subscriber-DN	レルムの DN	Oracle Internet Directory のレルム・エントリ	非推奨。Oracle Internet Directory でのユーザー検索には GUID ヘッダーを使用してください。[2]

[1] 廃止予定。次のリリースで廃止されます。
[2] 廃止予定。次のリリースで廃止されます。

用語集

dads.conf

データベース・アクセス記述子の構成に使用する Oracle HTTP Server 上のファイル。

GET

ログイン資格証明がログイン URL の一部として送信される認証方式。

httpd.conf

Oracle HTTP Server の構成に使用するファイル。

HTTP レスポンス・ヘッダー（HTTP response headers）

シングル・サインオン・アプリケーションに埋め込まれたデータ。mod-osso を使用する Single Sign-On Server に、強制認証やシングル・サインオフなどのアクションを実行するよう指示する。このようなヘッダーがない場合、アプリケーションは Single Sign-On SDK を使用してシングル・サインオン・システムと対話する必要がある。

mod_oc4j

Web リクエストを OC4J エンジンに送信する Oracle HTTP モジュール。

mod_osso

Oracle HTTP Server 上のモジュール。これにより、OracleAS Single Sign-On で保護されているアプリケーションは、ユーザーが Single Sign-On Server にログインした後、ユーザー名とパスワードのかわりに HTTP ヘッダーを受け入れることが可能になる。これらのヘッダーの値は、mod_osso Cookie に格納される。

mod_osso Cookie

HTTP Server に格納されているユーザー・データ。Cookie はユーザーの認証時に作成される。同じユーザーが別のアプリケーションを要求した場合、Web サーバーはシングル・サインオン Cookie ではなく mod_osso Cookie 内の情報を使用して、そのユーザーをアプリケーションにログインさせる。この機能により、サーバーの応答時間が短縮される。

OC4J（Oracle Containers for J2EE）

Java2 Enterprise Edition 向けの軽量かつスケーラブルなコンテナ。

Oracle HTTP Server

Hypertext Transfer Protocol を使用する Web トランザクションを処理するソフトウェア。Oracle 製品には、Apache Group が開発した HTTP ソフトウェアが使用されている。

POST

ログイン資格証明がログイン・フォームの本体として送信される認証方式。

Single Sign-On SDK

パートナ・アプリケーションのシングル・サインオンを有効にする API。SDK は、PL/SQL API、Java API、およびこれらの API の実装方法を示すサンプル・コードで構成されている。

Single Sign-On Server

ユーザーが経費報告、メール、福利厚生情報などのシングル・サインオン対応のアプリケーションに安全にログインできるようにするプログラム・ロジック。

外部アプリケーション（external application）

Single Sign-On Server に認証機能を委任しないアプリケーション。かわりに HTML ログイン・フォームを表示して、アプリケーションにおけるユーザー名とパスワードの入力を求める。ユーザーは、最初のログイン時にサーバーでこれらの資格証明を取得するように選択できるため、2 回目以降は、アプリケーションに透過的にログインできる。

強制認証（forced authentication）

ユーザーが事前定義された時間アイドル状態だった場合、ユーザーに再認証を強制する機能。OracleAS Single Sign-On では、グローバル・ユーザーの非アクティブ・タイムアウトを指定できる。この機能は、セキュリティに敏感なアプリケーションがインストールされている場合に使用する。

シングル・サインオフ（single sign-off）

シングル・サインオン・セッションを終了して、すべてのアクティブなパートナ・アプリケーションから同時にログアウトするプロセス。作業中のアプリケーションからログアウトすると、シングル・サインオフが実行される。

ダイナミック・ディレクティブ（dynamic directive）

特殊なエラー・コードを使用する HTTP レスポンス・ヘッダー。これを使用すると、シングル・サインオン・プロトコルを実装しなくても、アプリケーションはシングル・サインオン機能を利用できる。

データベース・アクセス記述子（database access descriptor）

Single Sign-On スキーマなどの特定の OracleAS コンポーネントに関するデータベース接続情報。

パートナ・アプリケーション（partner application）

Single Sign-On Server に認証機能を委任するアプリケーション。このようなアプリケーションでは、mod_osso ヘッダーを受け入れたり、ユーザーをサーバー自身にリダイレクトできるので、ユーザーを再認証する必要がない。ユーザー自身をリダイレクトするには、アプリケーションを Single Sign-On SDK に統合する必要がある。

索引

C

CREATE_ENABLER_CONFIG プロシージャ， A-6

D

DECRYPT_COOKIE ファンクション， A-10
DELETE_ENABLER_CONFIG プロシージャ， A-9

E

ENCRYPT_COOKIE ファンクション， A-10

G

GENERATE_REDIRECT ファンクション， A-2
GET_ENABLER_CONFIG プロシージャ， A-5
GET 認証方式， 2-11

H

HTTP ヘッダー， D-2

J

Java Server Pages,「JSP」を参照
Java パートナ・アプリケーション
  SDK 使用， B-3 〜 B-4
  静的な保護， 2-6, 2-7
  動的な保護， 2-7 〜 2-11
Java パートナ・アプリケーション、静的な保護， 2-6
JSP， B-4

M

mod_osso
  Single Sign-On SDK との比較， 1-2
  サンプル・アプリケーション， 2-4 〜 2-11
  定義， 1-2
  統合方法， 2-2
  利点， 1-2
mod_osso Cookie， 2-12

P

PARSE_URL_COOKIE プロシージャ， A-4
PL/SQL API,「Single Sign-On SDK」を参照
POST 認証方式， 2-11

S

Single Sign-On SDK
  Java API， A-12
  mod_osso との比較， 1-2
  PL/SQL API
    CREATE_ENABLER_CONFIG， A-6
    DECRYPT_COOKIE， A-10
    DELETE_ENABLER_CONFIG， A-9
    ENCRYPT_COOKIE， A-10
    GENERATE_REDIRECT， A-2
    GET_ENABLER_CONFIG， A-5
    PARSE_URL_COOKIE， A-4
    UPDATE_ENABLER_CONFIG， A-8
  非推奨， 1-2
  表
    WWSEC_ENABLER_CONFIG_INFO$， A-11
    WWSEC_SSO_LOG$， A-11

U

UPDATE_ENABLER_CONFIG プロシージャ， A-8
URL、保護， 2-2, 2-3

W

WWSEC_ENABLER_CONFIG_INFO$ 表， A-11
WWSEC_SSO_LOG$ 表， A-11

あ

アプリケーション・セッション Cookie
　クリア， 2-12
　コーディング， 2-12
アプリケーションのログアウト， 2-14
アプリケーションのログイン， 2-12 ～ 2-14

か

外部アプリケーション， 1-1
強制認証， 2-10, 2-14
グローバル・ユーザーの非アクティブ・タイムアウト
　， 2-11
コード例
　アプリケーションのログイン， 2-12 ～ 2-14
　強制認証， 2-10, 2-14
　シングル・サインオフ， 2-9, 2-10
　認証， 2-8, 2-9

さ

サーブレット
　SDK， B-3
　静的な保護， 2-6, 2-7
　動的な保護， 2-8 ～ 2-11
シングル・サインオフ， 2-9, 2-10
スタティック・ディレクティブ
　記述， 2-2
　定義， 2-2
成功時の URL， C-2

た

ダイナミック・ディレクティブ
　一般的なタイプ， 2-3
　サポートするプログラミング言語， 2-3
　定義， 2-3

な

認証、簡易， 2-8

は

パートナ・アプリケーション
　概要， 1-1
　追加， C-2, C-3
　編集， C-3, C-4
パートナ・アプリケーション、作成
　JSP の例， B-4
　PL/SQL の例， B-2
　サーブレットの例， B-4
「パートナ・アプリケーションの管理」ページ， C-1

や

ユーザー属性， D-2

ら

例外， A-12