WebSphere Portalにおける シングル・サインオン実現のための指針

複数のシステムへのアクセスの簡便化を実現するため、ポータル・サーバーが今、注目を集めています。ポータル・サーバーの利点の一つである「統一されたユーザー・インターフェースからの情報への一元アクセス」を実現するためには、さまざまなシステムへのシングル・サインオンを可能とするシステムを構築する必要があります。ところがポータル・サーバーから連携するバックエンド・サーバーには多くの種類があるので、最適な配置方法、接続の仕方など、さまざまな考慮点があります。本論文では、WebSphere® Portalを使用したシステム構築の際に、シングル・サインオンを中心としたシステムの実現可能性に注目します。そこで、さまざまな構成パターンの特徴を取り上げ、要件に応じた最適な構成パターンの選択の指針について述べていきます。

日本アイ・ビー・エム システムズ・エンジニアリング株式会社
Web & トランザクションシステム部
ITスペシャリスト
IT Specialist
Web & Transaction Systems
IBM Japan Systems Engineering Co.,Ltd.

神野 稚子 Wakako Jinno

[プロフィール]
1997年、日本アイ・ビー・エム入社後、日本アイ・ビー・エム システムズ・エンジニアリングに出向し、Web関連のプロジェクトに参画。1999年よりWebSphere Application Serverを担当し、研修開発やプロジェクト参画などを通じWebシステム構築に携わっている。

日本アイ・ビー・エム システムズ・エンジニアリング株式会社
ITソリューション・センター ポータル&コラボレーション・システム部
グループリーダー
Group Leader
Portal & Collaboration Systems, IT Solution Center
IBM Japan Systems Engineering Co.,Ltd.

青山 桐子 Kiriko Aoyama

[プロフィール]
1990年、日本アイ・ビー・エム入社。通信メディアと保険インダストリーでAIX関連のシステム提案・構築に従事した。2000年に日本アイ・ビー・エム システムズ・エンジニアリングに異動。WebSphere Portal製品発表と同時に担当し、提案サポートを始めたプロジェクトに参加して、実際のポータル・システム構築に携わっている。

Guidelines for Realization of Single Sign-on in WebSphere Portals

Portal servers are currently provoking a great deal of interest due to their potential for enabling easier access to several systems. In order to realize the "unitary access to information from a unified user interface" which is one of the main advantages of portal servers, it's essential to construct a system which enables single sign-on to various systems. But since there are many types of back-end server that link up from portal servers, attention must be directed to a whole range of questions such as the most appropriate method of layout and methods of connection. In this paper I focus on the potential for realization of systems centering on single sign-on in the case of construction of systems using WebSphere® portals. I take a look at the features of various configuration patterns and refer to guidelines for the selection of the most appropriate configuration patterns in accordance with the conditions.

IBM PROfessionals' PAPERS ❻

1. はじめに

インターネットやイントラネットの急速な普及により、企業内での情報共有にWeb技術を適用したシステムが利用されるようになってきました。反面、企業内の部門や機能ごとにシステムが乱立し、それぞれのシステムのユーザー・インターフェースを通じて、異なるユーザー認証を行うようになってしまいました。その結果、ユーザーは情報を探し出したり、ユーザーIDやパスワードを管理したりするのに苦労しています。

こうした状況の中、ユーザーへの統一された窓口を提供するポータル技術が脚光を浴びています。この章では、ポータル技術の概要を紹介していきます。同時に、ポータルに必要な機能、技術についても述べていきます。

1.1. ポータルに求められる機能

IT(Information Technology：情報技術)業界における、「ポータル」には、単にほかのシステムへの「入り口」としての機能だけでなく、ユーザーの特性や嗜好(しこう)に応じたサービスやコンテンツを提供するための「パーソナライゼーション(Personalization)」機能が求められています。

本論文では、「ポータル(Portal)」に求められる機能を、IBMのポータル製品「WebSphere® Portal」では、どのように実現しているのかを述べていきます。WebSphere Portalは、一画面のフレームの枠内の内容の一つひとつをポートレットと呼ばれるサーブレットのサブクラスで表示しています。また、ユーザーごと、グループごとに表示するポートレットが選択可能です。同じポートレットであっても個人の情報によって、表示させる内容を変えることもできるのです。

この機能を実現するためには、通常、使用するユーザーを特定する必要がありますが、WebSphere Portalでは、この仕組みをWebSphere Securityの機能を用いて実現しています。

また、さまざまな情報への「入り口」として動作するためには、情報の提供元である各種のバックエンド・サーバーにアクセスする必要があります。

これらのバックエンド・サーバーは、既に構築され稼働しているケースがほとんどですが、それぞれに異なる認証方式を採用しているケースも少なくありません。ITシステムが便利になり、新しいアプリケーションが増えるにつれて、一人のユーザーが管理しなければいけないユーザーIDやパスワードの数が増えていくという経験をされている方も多いでしょう。システムとしてのセキュリティー・レベルをいくら高くしても、実際に使用するユーザーにとって使いやすいシステムでない限り、優れたシステムとはいえません。現に、複数のユーザーIDやパスワードを覚えきれずに、紙に書いて端末の前に張っているユーザーを見掛けることすらあります。これでは、せっかくのセキュリティー・システムが台無しになってしまいます。

このような危険性を減らし、ユーザーの利便性・効率を上げるためにも、ポータルにおいてはシングル・サインオンが強く求められる傾向にあるのです。

1.2. WebSphere Portalのシングル・サインオン方針

では、WebSphere Portalは、どういった方針でシングル・サインオンを実現しているのでしょうか。

一般的なシングル・サインオンの実現方法には、「リバース・プロキシー・タイプ」と「アクセス・チケット・タイプ」と呼ばれる二つのアプローチがあります。

「リバース・プロキシー・タイプ」のシングル・サインオンでは、リバース・プロキシー上で、ユーザーの認証とアクセス権の検査が行われます。これによって許可されたHTTP(Hypertext Transfer Protocol)リクエストだけが、バックエンドのWebサーバーに転送されます。リバース・プロキシーとWebサーバーの間では、通常のHTTPによる通信が行われるため、各Webサーバー上では、シングル・サインオンのための特別な対応の必要がなく、非常に汎用性が高いといえます。後に紹介する、Tivoli® Access Manager(以下、Access Manager)は、リバース・プロキシー・タイプの認証を採用しています。

「アクセス・チケット・タイプ」では、認証サーバーにおいてユーザーの認証とアクセス権の検査を行います。許可された場合は、認証情報を埋め込んだ「アクセス・チケット」が発行されます。その「アクセス・チケット」を、クッキーやURL(Uniform Resource Locator)を使用してサーバー間で持ち回ることによって、シングル・サインオンを実現します。この方法では、アクセス・チケットを受け取って、アクセス権の検査を行うため、各Webサーバー上に特別な仕組みが必要となります。

WebSphere Application Server(以下、WebSphere AS)の

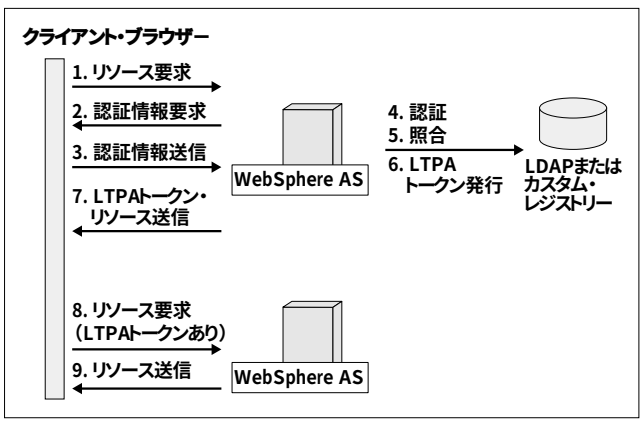

図1. LTPA認証処理の流れ

LTPA(Lightweight Third Party Authen-tication)認証は、アクセス・チケット・タイプの方法を採用しています。WebSphere Portalは、ポータルが稼働しているWebSphere ASのセキュリティーを使用することを前提としています。WebSphere ASでのシングル・サインオンは、LTPAトークンというクッキーを用いることでサーバー間で認証情報を受け渡すアクセス・チケット方式です。図1に、LTPA認証処理の流れを示します。

- LTPAによるシングル・サインオンを行おうとするユーザーは、「LTPAキー」を共有します。「LTPAキー」には対になるパブリック・キー、プライベート・キー、3DES共通鍵が含まれます。
- 保護されたリソースへのアクセスが発生するとWebSphere ASは認証を行い、認証をパスするとLTPAトークン(クッキー)が発行されます。
- LTPAトークンとは「ユーザーID+LTPAトークンが失効するタイム・スタンプとプライベート・キーで作成された署名がBase64でエンコードされた物」を、さらに3DES(Triple Data Encryption Standard)で暗号化したものです。クッキーにするときに、それをBase64でエンコードします。
- さらに、ユーザーが、もう一つのWebSphere ASにアクセスしようとするとき、LTPAトークンとともにリクエストを送信します。

WebSphere ASは送られたLTPAトークンをデコードし、パブリック・キーにより、署名が有効なものであるかの確認後に、ユーザーが、そのリソースにアクセスする権限を持っているのならば、リソースを送信します。

また、クッキーの仕組みを使用しているため、シングル・サインオンを実現したいサーバー間で、同じドメイン名が必要になります(例:server1.domain.co.jp とserver2.domain.co.jp)。

LTPA による認証は、WebSphere Portal (WebSphere AS)のほかに、Domino™でもサポートされています。これらの間ではLTPAを使用したシングル・サインオンが実現可能です。

しかし、LTPAを採用していないサーバーとポータルを連携したいという要件は多数あります。連携したいサーバーがどのような認証方式を採用しているのか、配置はどうするかなどの条件によって連携方針もさまざまです。では、こういった要件には、どのように対応すればよいのでしょうか。

2. WebSphere Portal構成パターン

WebSphere Portalの構成を考える上で重要な点は、バックエンド・サーバーとの連携方法と、ユーザーIDやパスワードの格納個所をどこに配置するかの2点です。この章では、ポータルやバックエンド・サーバーで採用している認証方針を踏まえつつ、この2点についての考慮点を述べていきます。なお、認証を行うために必要なユーザーIDとパスワードの格納個所を、本論文では「ユーザー・レジストリー」という言葉を用いて表します。

2.1. WebSphere Portalとセキュリティー

2.1.1. セキュリティーの二つの局面

一般に、アプリケーションのセキュリティーには、「認証」と「認可」という局面があると考えられます。

- 認証(Authentication)

リソースを要求するユーザーが「だれであるのか」、「確かに本人であるのか」などを確認することを指します。最も一般的な認証の手法は、ユーザーIDとパスワードの照合によって審査する方法です。

- 認可/アクセス制御(Authorization)

認証されたユーザーが、要求するリソースにアクセスする権限を持っているかどうかを判定します。WebSphere AS(WebSphere Portal)が準拠しているJ2EE(Java™ 2 Platform Enterprise Edition)のセキュリティーにおいてアクセス制御の対象となるリソースは、次の二つです。Webクライントの場合は、HTTPのメソッド(GET/POST/PUT/DELETE)です。EJB(Enterprise JavaBeans)クライアントの場合は、EJBのメソッドになります。

WebSphere AS(J2EE)のセキュリティーのフレームワークでは、アクセス制御はHTTPかEJBのメソッド単位に行われます。しかし、実際のアプリケーションを考えると、メソッド単位では要件を満たし切れないことが多くあります。例えば、データの内容によってユーザーに見せる内容を制限したりすることは、メソッド・レベルのアクセス制御では不可能です。また、ある処理ができるユーザーを制限するといった場合でも、その処理が、EJBのメソッドで実装されている必要があっても、アクセス制御のためだけにEJBを使用するということはあまり現実的ではありません。多くの場合は、ユーザーIDを取得して、アクセス制御をするロジックをアプリケーションで実装することになります。WebSphere Portalでも認証部分は、WebSphere Securityを使用しています。しかし、アクセス制御やユーザーごとの表示内容の管理は、ポータルが提供している機能での実装となっています。WebSphere Portalは、J2EEのロールをサポートしておらず、ポータル内のリソースに当たるプレースやページ、ポートレットなどに対するアクセス制御は、「アクセス管理ポートレット」を通じて行われます。また、アクセス制御を外部のセキュリティー・マネジメント・ソフトウェアに委託することも可能です。

Access Managerおよび、Site Minderへのインターフェースが提供されています。

このように、一般にアクセス制御については、ほとんどのシステムが個々に対応しています。本論文で述べるシングル・サインオンの対象も、「認可」ではなく「認証」についての連携です。

2.1.2. ポータルで選択可能な認証形態

WebSphere Portalが使用しているWebSphere ASのセキュリティーでは、ユーザーが自分自身に関する情報（ユーザーIDやパスワードなど）を提示する方法として次の4種類を提供しています。

（1）なし（None）

ユーザーがユーザーIDやパスワードなどの入力を求められません。リソースがセキュリティー機能によって保護されている場合は、ユーザーはリソースにアクセスすることができません。

（2）基本（Basic）認証

HTTP 1.0／1.1の仕様で定められた認証方法です。ユーザーがブラウザーのポップ・アップ画面から入力したユーザーIDとパスワードで認証が行われます。いったん入力したユーザーIDとパスワードは、ブラウザーにキャッシュされ、リクエストごとにHTTPヘッダーに含められ、自動的にWebサーバーに送信されます。ユーザーIDとパスワードは、Base64でエンコーディングされますが、Base64は簡単にデコードできるので、セキュリティーを確保するためには、SSL（Secure Socket Layer）を併用した通信の暗号化が必須となります。

（3）SSLクライアント証明書（Certificate）

クライアント・マシンに組み込まれたX.509形式の証明書に基づいて、ユーザーの認証が行われます。クライアント側には、各自のX.509証明書とプライベート鍵を所有している必要があります。これはSSLによる高度な認証です。ネットワーク上に流れるデータを盗聴されても“なりすまし”の危険がありません。非常に安全な認証方式です。

（4）フォーム（Form）認証

ログイン・ページに、HTML（Hypertext Markup Language）フォームを使用する認証です。ログイン・ページは、HTMLなので自由にカスタマイズすることができます。ユーザーの入力したユーザーIDおよびパスワードは、アプリケーション・データとして、ブラウザーかサーバーに送信されます。認証が成功すると、LTPAトークンという認証情報が入ったクッキーが発行されます。WebSphere Portalでは、必ずユーザーの確認が必要です。そのため実質認証「なし」を除いた、「基本認証」「SSLクライアント証明書」「フォーム認証（LTPA）」の3タイプが考えられます。

2.1.3. バックエンド・サーバーの認証形態

上記で紹介したWebSphere ASのセキュリティーがサポートしている4種類の認証タイプのほかに、バックエンド・サーバー側では、製品やアプリケーションで独自の認証方式を採用している場合が考えられます。ポータルでは、「基本認証」「SSLクライアント証明書」「フォーム認証（LTPA）」の3タイプが考えられるので、バックエンド・サーバーでは、「認証なし」「基本認証」「SSLクライアント証明書」「フォーム認証（LTPA）」「フォーム認証（独自認証）」の4タイプが考えられます。

2.1.4. 認証タイプとヘッダーに付加される情報

クライアント・ブラウザーからポータルへアクセスがあったとき、ポータルが使用しているそれぞれの認証タイプにおいて、クライアント・ブラウザーとポータル間で、やり取りされる情報が異なります。そのHTTPヘッダーに、付加される情報を表1にまとめてみました。

基本的に、認証時にヘッダーに付加されるものを（BAヘッダー、LTPAトークン）、そのサーバーに受け渡せればシングル・サインオンが実現可能です。それができない場合は、認証時に要求されるもの（ユーザーIDやパスワードなどのフォーム入力データ、SSLクライアント証明書）を受け渡す必要があります。

2.1.5. 二つのアプリケーション実装方針

WebSphere Portalには大きく分けて二つのアプリケーション実装方針があります。これらによって、シングル・サインオンの実現可能性も異なってきます。

（1）ネイティブ・ポートレットを使う方針

バックエンド・アプリケーションをポートレットでキックし、ポートレットが結果を受けて整形して表示します。これにはバックエンドのアプリケーションとのインターフェース決めが必要です。また、既存バックエンド・アプリケーションに手を入れなければならない場合もあります。

HTTPリクエストのフローは図2のようになります。ポートレットで、ほかのサーバーの内容を取り込むので、クライアントからはポータルを経由して、そのサーバーにHTTPリクエストが飛びます。

表1. HTTPヘッダーに付加される情報

認証のタイプ	レジストリー	ヘッダーに付加される情報
なし	なし	なし
基本認証	LDAP／Custom	LTPAトークン、BAヘッダー
	Native OS	BAヘッダー
フォーム認証	LDAP／Custom	LTPAトークン
SSLクライアント証明書	LDAP／Custom	なし（証明書の情報）※

※LTPAトークンは発行されない。クライアント証明書のユーザーIDを信頼するため、WebSphere ASでは認証は起こらず、アクセス権限のチェックのみ行われる。

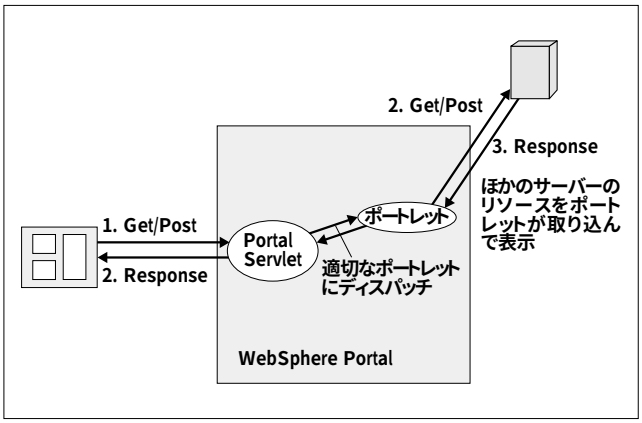

図2. ネイティブ・ポートレットを使う場合のHTTPリクエストのフロー

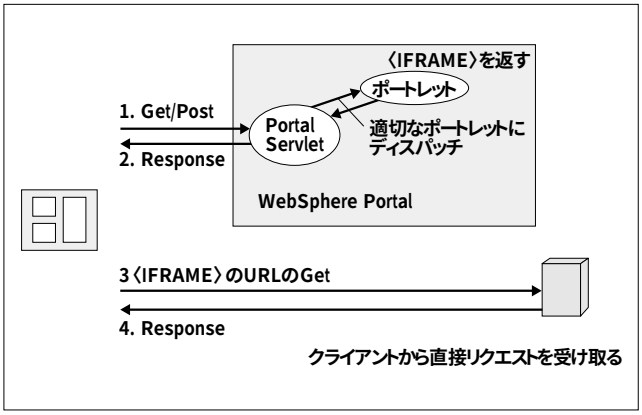

図3. FRAME対応ポートレットのHTTPリクエストのフロー

(2)IFRAME対応ポートレット(<IFRAME>タグ)を使う方針

<IFRAME>とは、HTML4.0で定義されているタグで、Inline Frameの略です。Internet Explorer3.0以上、Netscape Communicator6.0以上でサポートされています。

<IFRAME>を使用すると、<FRAMESET>のように、ウィンドウを水平・垂直に分割する形式だけではなく、ウィンドウの中に独立して表示される形式のインラインのフレームを作成することができます。IFRAMEタグのsrc属性に表示したい内容のURLを指定すればよいので、手軽に使われている方法です。HTTPリクエストのフローは、図3のようになります。

2.2. WebSphere Portalからのシングル・サインオン実現方法

この章では、シングル・サインオンの実現方法の観点から、WebSphere Portalを用いてポータル・サイトを構築するための構成の選択基準を探ります。

2.2.1. ポータルからの認証情報受け渡し方法とその考慮点

シングル・サインオンを実現する場合は、認証された情報を「どのように渡すか」が重要になってきます。シングル・サインオンを行うとき、認証情報をHTTPヘッダーに入れて送る必要があります。そこで、シングル・サイン対象のサーバーで採用している認証方法や、HTTPフローの違いにより認証情報をどのように渡すかなどを考えていく必要があります。この章では、認証形態やポータルでのアプリケーション実装方針により、それぞれに適したシングル・サインオンの方法を考えていきます。

バックエンド・サーバーで、「認証なし」「基本認証」「SSLクライアント証明書」「フォーム認証(LTPA)」「フォーム認証(独自認証)」などの認証タイプを採用している場合は、二つの実装方針の認証情報があります。まずは、それらの認証情報をバックエンド・サーバーに受け渡しする方法について述べていきます。

(1)バックエンド・サーバーが認証なしの場合

ネイティブ・ポートレットを使用する場合は、認証情報は受け渡す必要はありません。

IFRAMEポートレットを使用する場合は、HTTPリクエストがポータルを経由せず、直接バックエンド・サーバーに届きます。保護したいコンテンツが認証なしのサーバーに配置されている場合は、認証サーバーの採用を検討するか、バックエンド・サーバーをポータルと連携しやすい方法(基本認証、LTPAなど)で、保護を考える必要があります。

(2)バックエンド・サーバーが基本認証(Basic Authentication)で保護されている場合

バックエンド・サーバーが基本認証で保護されている場合は、バックエンド・サーバーは、BAヘッダー(Authorizationヘッダー)の中に入っているBase64でエンコードされたユーザーIDとパスワードを利用して認証を行います。

ネイティブ・ポートレットを使用する場合は、ポータルからBA(Authorization)ヘッダーをバックエンド・サーバーに渡す必要があります。ポータルの機能としては、BAヘッダーを転送する機能は標準では持っていないので、そのためのポートレットを作成しなければなりません。また、後述するCredential Vault機能を使用することも可能です。ポータルで認証したユーザーIDとパスワードを基に、BAヘッダーを組み立てることで後ろに渡しています。

IFRAMEポートレットを使用する場合は、HTTPリクエストがポータルを経由せずに、直接、バックエンド・サーバーに届きます。その時点で、基本認証が発生します。ポータルだけのシングル・サインオンは不可能です。後述の認証プロキシーの採用を検討する必要があります。

(3)バックエンド・サーバーが、SSLクライアント証明書による認証を行っている場合

ネイティブ・ポートレットを使用する場合は、クライアント・ブラウザーからのSSL通信は、ポータルでいったん終了してしまうので、バックエンド・サーバーまでクライアント証明書の情報が届きません。そのままポータルーバックエンド・サーバー間のクライ

アント証明書による認証を行おうとすると、ポータル・サーバーにクライアント証明書を導入して通信することになります。この場合は、バックエンドには、ポータル・サーバーに導入されたクライアント証明書の情報が届き、ユーザー個別のクライアント証明書の情報は届きませんので、ユーザーの情報は、別途ポートレットから送られたヘッダー情報を基にアプリケーションで取り出す必要があります。その結果、バックエンドでSSLクライアント証明書の情報を基にユーザーの識別を行い、アプリケーションの挙動を変えるといった実装をしている場合は、ポータル経由では正常に行えなくなります。SSLクライアント証明書による認証に変わる方法を検討するか、アプリケーションに変更を加えるための対応が必要になってきます。

IFRAMEポートレットを使用する場合は、クライアント・ブラウザーから異なるサーバーにログオンすると同じ動きになってしまいます。そこでSSLのハンド・シェイクを、サーバーの数だけ発生させて、ユーザーからは証明書をその数だけ提示させる必要が出てきます。また、ポータルとバックエンド・サーバーで同じルート証明書を持っているサーバー証明書を使用します。同じクライアント証明書で認証されるようにすれば、ある意味シングル・サインオンのような動きをすると考えることができます。

(4)バックエンド・サーバーでLTPA認証が採用されている場合

ポータルもLTPA認証を選択し、LTPAトークンによるシングル・サインオンを行います。ポータル側では、LTPAの環境下でさえあれば、「基本認証」「フォーム認証」など、いずれの場合も選択可能です。

ネイティブ・ポートレットを使用する場合は、URLレベルでアクセスすると、LTPAトークンがバックエンド・サーバーに送られ、それを基にシングル・サインオンが行われます。LTPAのシングル・サインオンはクッキーを利用しているため、ポータルとシングル・サインオンを行いたいサーバーは、同じドメインに属している必要があります。ポートレットからJavaアプリケーションとしてアクセスする場合には、CORBA credentialからLTPAトークンを抜き出して、バックエンド・サーバーへのアクセスに使用することができます。LTPAトークンを抜き出したり、ユーザーIDとパスワードを保管するためには、WebSphere PortalのCredential Vault機能を使用することができます。

IFRAMEポートレットを使用する場合は、上記のURLレベルのアクセスと同様です。ポータルを経由せずに、直接バックエンド・サーバーにアクセスがあったとしても、バックエンド・サーバーで設定された認証が発生するので、LTPAトークンを発行することになります。

(5)バックエンド・サーバーで独自の認証が採用されている場合

ネイティブ・ポートレットを使用する場合は、ポートレットを作成し、バックエンドのログイン・フォームや独自認証の形式に合うように、ユーザーIDとパスワードをポストで送り込む仕組みをつくる必要があります。これには、WebSphere Portalが提供するCredential Vaultを使用することができます。バックエンド・サーバーにログインするために必要な情報(ユーザーIDやパスワードも含む)をバリュー・サービスのレジストリーから取り出して、ポートレットからアクセスするためのものです。

IFRAMEポートレットを使用すると、ポータルを経由せずに、直接、バックエンド・サーバーにアクセスしたときに認証が発生する場合があります。そのため、ポータルだけのシングル・サインオンは不可能です。後述の認証プロキシーの採用を検討する必要があります。

2.3. ユーザー・レジストリー連携

ポータルで認証を行う際、ユーザーIDとパスワードを、ユーザー・レジストリーを利用して照合します。そのときにポータルから連携したいバックエンド・サーバーも、既にユーザー・レジストリーを持っている場合が多くあります。管理や運用面を考えると、当然、ユーザー・レジストリーは統合できた方がよいでしょう。ここでは、どういった場合に統合することができるのかを考察していきます。

2.3.1. ポータルで使用可能なユーザー・レジストリー

WebSphere Portal (WebSphere AS)で認証を行う際に選択できるユーザー・レジストリーは、LDAP(Lightweight Directory Access Protocol)ディレクトリー・サーバーとカスタム・レジストリーの二つがあります。LDAPディレクトリー・サーバーは、LDAPに準拠したディレクトリー・サーバーを使用します。IBM SecureWay® Directoryのほか、複数のLDAP製品が使用可能です。カスタム・レジストリーは、提供されているAPI(Application Program Interface)を拡張することにより、データベースなど、任意のレジストリーを使用できます。

2.3.2. ユーザー・レジストリー共有の検討

- ユーザー・レジストリーを共有する(できる)場合

バックエンド・サーバーが、すべてのユーザー・レジストリーとしてWebSphere ASがサポートしているLDAPを使用している場合は、ポータルも、そのLDAPをユーザー・レジストリーとして使用することできます。例えば、バックエンドが複数のドミノ間で、公開アドレス帳を共有している場合が、これに当たります。

- ユーザー・レジストリーを共有しない(できない)場合

しかし、バックエンド・サーバーで、WebSphere ASから使用できない形式のユーザー・レジストリーを使用していたり、個々

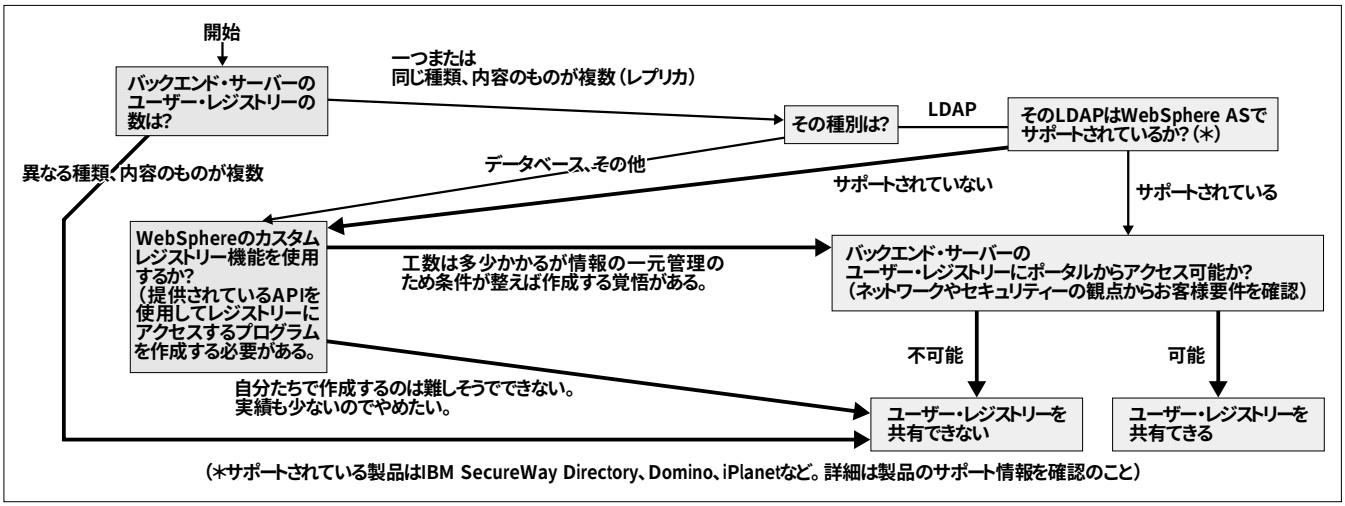

図4. お客様要件による判断基準のフロー・チャート

のバックエンド・サーバーで異なるユーザー・レジストリーを使用していたりすると、ポータルからの共有はできません。また、ユーザーIDやパスワードが、それぞれ異なるシステムと連携をしている場合は、工夫をする必要があります。もちろん、情報の一元管理といった観点からは、ユーザー・レジストリーは共有化できた方がよいでしょう。その判断基準は、上に述べた既存のユーザー・レジストリーが、WebSphereでサポートされているかどうかです。「されていない場合」は、WebSphereのカスタム・レジストリーを利用できるかどうかです。図4に、ほかのお客様要件による判断の目安を、フロー・チャートにまとめてみました。

2.3.3. ユーザー・レジストリーが共有できない場合の考慮点

さて、ユーザー・レジストリーが共有できないと判明した場合でも、どうにかしてポータルでのシングル・サインオンを実現しなければならないことがあります。そのためには、ユーザーがポータルのログインの際に使用するユーザーIDとパスワードが、ポータルのユーザー・レジストリーに入っている必要があります。この際、バックエンド・システムとポータルでユーザーIDとパスワードが異なってしまうと、ユーザーは、直接バックエンド・サーバーへログインする場合と、ポータルへログインする場合の、2組のユーザーIDとパスワードを管理する必要が出てきてしまい、それほどポータルのシングル・サインオンの恩恵を受けられないことになります。レジストリーが物理的に共有できない場合は、できれば、ユーザーIDとパスワードは複数システム間で統一したいものです。何らかの理由でユーザーIDとパスワードが複数システムで別々になってしまう場合は、ポータルで認証するユーザーIDとパスワードと、バックエンドへ流すユーザーIDとパスワードをマップするテーブルを作成して、管理する必要があります。こういう仕組みは、ポータル標準にはないため、ユーザーが独自にそういった仕組みをつくり込まなければなりません。そのためこういった仕組みを作成するために、開発・テストにかなりの工数を要することになります。

2.4. 認証プロキシーの使用を考える

今まで述べてきた構成は、ポータルで認証を行う場合でしたが、ポータルのフロントに認証プロキシーを置くという形態も考えられます。この章は、認証プロキシーを置く場合についての構成パターンと考慮点を述べていきます。

2.4.1. 採用可能な認証プロキシー

ポータルは自身で認証する構成以外に、認証プロキシーと連携する機能も提供しています。ポータルで想定している認証プロキシーは、Access Manager 3.9（旧Tivoli Policy Director）とSiteMinderです。認証プロキシーとは、Trust Association Interface（TAI）というインターフェースを使用した連携が想定されています。Access ManagerやSite Minderに、それぞれ幾つかのバージョンに対応したTrust Association Interfaceが用意されていますが、その連携方法には、Trust Association Interfaceを使う以外にも幾つか方法があります。要件にふさわしい方法を選択しましょう。

次からは、Access Managerとの連携を中心に、可能な連携パターンを検討していきます。

2.4.2. Access ManagerとPortal構成パターン

ポータルのフロントでの認証サーバーとして想定されているAccess Managerは、認証とアクセス制御を集中管理するための製品です。Access Managerの中でも、最も重要なコンポーネントであるWebSEALは、認証と認可のサービスを提供することが可能なリバース・プロキシーです。

WebSEALは、後ろに配置したWebサーバーを自分のURL

以下に階層的に管理し、認証と認可のサービスを提供します。

この「後ろに配置する」ということを「ジャンクション」するといいます。ジャンクションを作成する際に指定するさまざまなオプションにより、後ろに付加する情報を選択することができます。図5に代表的なジャンクション・オプションの概要を紹介します。

ポータルでは、「情報をバックエンドに送る」ためには専用のポートレットを作成する必要がありましたが、WebSEALの機能では、ジャンクション・オプションにより簡単に実現できるのです。

WebSphere Portalが、WebSphere Securityを使用していることを考えると、ポータルとAccess Managerの連携方法はそれほど多くありません。「基本認証による連携」「LTPAジャンクション」「Trust Association」などのうちのどれかということになります。

■ - b　オプション
BA（Authorization）ヘッダー関連のオプション。
• **-b supply**
BA（Authorization）ヘッダーに、WebSEALで認証されたユーザー名とダミーの固定パスワードを挿入し、リクエストを転送。
• **-b filter**
WebSEALで基本認証を行い、バックエンドで認証を行わない。HTTPヘッダーに、ユーザーID・パスワードを含めず（ユーザーID・パスワードを除去し）、リクエストを転送。
• **-b gso**
WebSEALで認証を行った後、設定しておいたGSOテーブルより、バックエンドに流すユーザーIDとパスワードを取得。BA（Authorization）ヘッダーに含めてリクエストを転送。

■ -A オプション
LTPAジャンクション
WebSEALで認証されたあと、LTPAトークンを作成して後ろに流すことができる。

■ -c オプション
認証したユーザー名やグループ、クレデンシャルなどをそれぞれiv_user、iv_group、iv_creds、iv_user_lというHTTPヘッダーで転送。

■ -S -f fsso.conf FSSO
バックエンド・サーバーがフォーム認証を採用している場合、そのフォームに合ったHTTPリクエストを転送することができる。

図5.　代表的なジャンクション・オプションの概要

(1) WebSEAL―ポータル間の基本認証による連携

これは最も実績のある構成です。ただし、WebSEAL-ポータル間でユーザーIDとパスワードがネットワーク上を流れてしまうという懸念があります。BA（Authorization）ヘッダーは、Base64でエンコードされているとはいえ、Base64は簡単にデコードできる形式です。この点については、お客様環境のセキュリティー要件を確認する必要があります。-b supplyで、ダミー・パスワードを流す場合は、この点はあまり心配ないように思えますが、WebSEALを経由せず直接ポータルにアクセスされる場合を考えると、ダミー・パスワードがばれてしまう危険性があります。こういった場合も、ネットワーク設定で回避できる場合が多くあります。回避できない場合は、SSLジャンクションの使用を検討するとよいでしょう。ただし、SSLを使用すればその分サーバーに負荷がかかるので、この点はセキュリティーとパフォーマンスとのトレード・オフになります。

(2) WebSEAL―ポータル間のLTPAによる連携

Access Managerの前のバージョンであるPolicy Director 3.8からサポートされるようになったLTPAジャンクションを使用した連携です。WebSphere ASでLTPA Keyを作成し、WebSEALにインポートすることによって使用可能です。ブラウザーにはLTPA トークンは送らず、WebSEAL側とバックエンド・サーバー間のみで、LTPA トークンのやり取りが行われます。ユーザーごとに、LTPA トークンが作成されるため、メモリー・リソースがWebSEALで消費されます。また、HTTPリクエストごとに毎回LTPAトークンが作成されるとパフォーマンス上に多大な影響を与えます。そのため、WebSEAL側でのLTPAキャッシュ設定はほぼ必須です。

(3) WebSEAL―ポータル間のTrust Association

Trust Associationとは、WebSphere ASのセキュリティー構成方法の一つです。認証プロキシーをフロントエンドに置き、そ

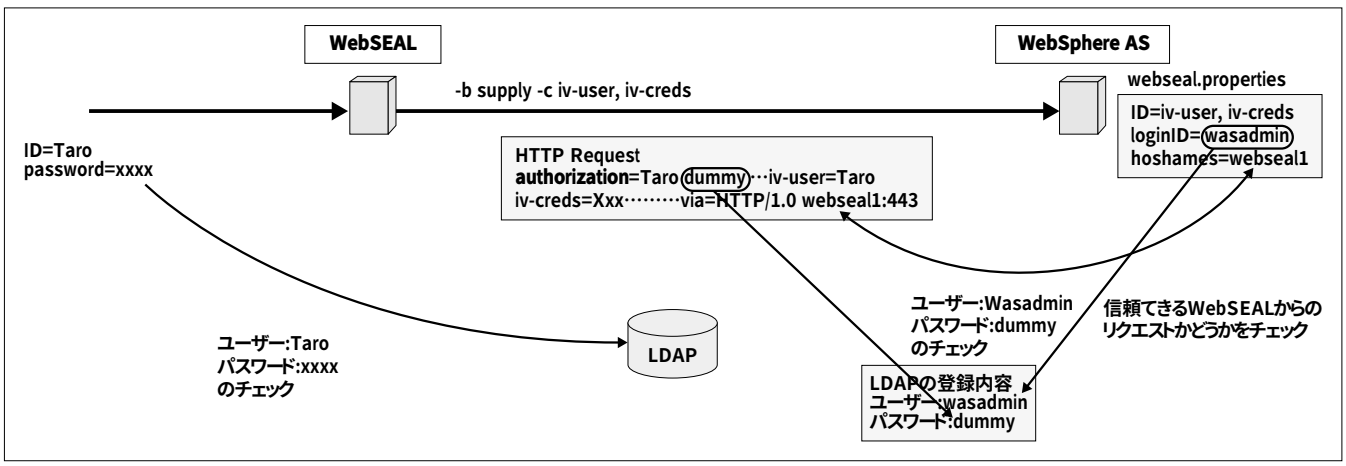

図6.　Trust Associationの基本的な動き

の認証プロキシー経由リクエストである場合は信頼します。「WebSphere ASでは認証（ユーザーID／パスワードのチェック）を行わない」という構成の場合は、このWebSphere Portalの構成が推奨されています。図6に、Trust Associationの基本的な動きを示します。

- WebSEAL（-c -b supplyの利用）
  1. クライアントからのユーザーIDとパスワードをWebSEALが受け取り認証を行います。
  2. HTTPヘッダーにユーザーIDとクレデンシャル、-b supplyでセットされたダミーのパスワードを付け加えてバックエンドのWebSphere ASに送ります。
- WebSphere AS
  1. 信頼したWebSEALからのリクエストか（viaヘッダーの内容とwebseal.propertiesに定義したホスト名の照合）。
  2. -b supplyのパスワード＋WAS側で事前定義したUIDのパスワードが一致しているかどうかをLDAPに照合します。
  3. 要求したヘッダー（iv-user, iv-creds）がHTTPリクエストの中に入っているかを検査します。
  4. クライアント・ユーザー（iv-userのユーザー名）が要求したリソースにアクセス可能かどうかを検査します。アクセス権限があることが確認されると、要求されたリソースは再びジャンクションを通ってクライアントに返されます。

WebSEALを経由せずに、直接WebSphere ASにリクエストがあった場合は、WebSphere ASに設定してある認証方法での認証が発生します。

これはWebSphere AS V4.02でTrust Associationのモジュールであるに変更があったからです。以前まではWebSEALの信頼性確認のために、WebSEALサーバーIDをBA（Authorization）ヘッダーに入れて送信する必要がありましたが、WebSEALの制限上、SSLジャンクションが必須でした。今回の変更により、-b supplyによるダミー・パスワードとプロパティー・ファイルに定義されたIDのパスワードの照合というより安全な方式に変更になり、ジャンクションもSSL必須ではなくなりました（もちろん、要件によりSSLを使用することも可能です）。以前までは、SSLを使用する際のパフォーマンス上の懸念から、Trust Associationの採用を躊躇（ちゅうちょ）してしまう場合が多かったのですが、今回の変更で、より採用しやすくなったといえます。

ダミー・パスワードがネットワーク上を流れる点は、基本認証で、-b supplyオプションを使用したときと一見同じように見えますが、「WebSEAL経由のリクエストでは認証は発生せずに、ポータルへアクセスが合った場合のみ発生する」という動きがあります。そのためユーザーがWebSEALを経由せずアクセスする場合は、リポジトリー内のユーザーごとの実パスワードを知る必要があります。つまり、実パスワードがネットワークを流れず、ネットワーク上を流れているダミー・パスワードでは、どこへも認証を通ることができないという点で安全性が高いといえます。

2.4.3. Access Manager ― ポータル連携サマリー

ここではポータルの連携方法の特徴をまとめます（表2）。送られるデータ量と、その内容（パスワードを送らないなど）による安全性など機能面だけを考えると、「Trust Association」「LTPA」「基本認証」の順でよいでしょう。ただし、LTPA、Trust Associationは機能が比較的新しいため、実績の多さは残念ながら逆の順番になっています。どちらを重視するかはプロジェクトでの判断にお任せしたいと思います。

2.4.4. 認証プロキシー採用の検討

- ネットワーク構成とシングル・サインオン

認証プロキシーを使った場合は、ポータルを含むアプリケーション・サーバーをDMZ（DeMilitarized Zone）の後ろの配置でできるというネットワーク構成上の利点があります。ポータル以外のサーバーへのリクエストに対しても、必ず認証プロキシーを通るので、ポータル以外のサーバーに対してもシングル・サインオンが容易に実現可能です。基本認証をサポートしていれば、ほとんどのバックエンドのWebサーバーとシングル・サインオンが実現可能です。

反面、WebSphere Portalだけの認証は、基本的にLTPAを使用していないサーバーとはシングル・サインオンできず、ポー

表2. 連携方法の特徴

＊ LTPAトークンは、ユーザーIDの長さにより多少変化するが約380バイト

	送られるデータについての考察			パフォーマンス	実績
	送られるデータ	データ量	安全性		
基本認証	BAヘッダー（ユーザーID／パスワード）	小	低め	バックエンドで認証が起こる分の負荷はかかる。	多い。
LTPA	LTPAトークン（ユーザーID+α）が暗号化されたもの。パスワードは含まれていない。	中(＊)	高い	LTPAの作成やデコードに多少負荷がかかる。キャッシュの使用で回避する。	LTPAジャンクションはPolicy Director3.8からの機能のため、まだ少ない。
Trust Association	BAヘッダー（ユーザーID／ダミー・パスワード）	小	高い	バックエンドで認証が起こらないので、パフォーマンスはよい。	Trust AssociationはWebSphere AS V3.52からの機能のため、まだ少ない。

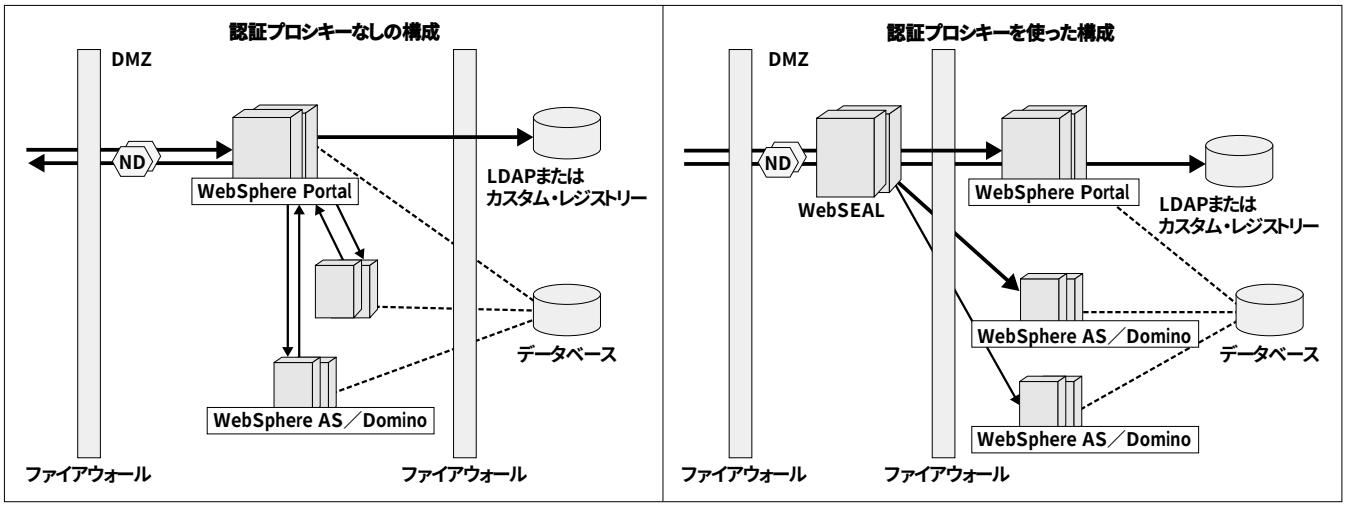

図7. 認証プロキシーの構成

トレットをつくり込む必要があります。逆に、ポータル連携したいサーバーがLTPAを使用できるものばかりで、LTPAによるシングル・サインオンだけで完結する場合のみ認証サーバーなしの構成が可能です（図7）。

・パスワード管理

WebSphere Portal（WebSphere AS）のセキュリティー機能だけを使用した場合は、パスワードの期限を設定したり、パスワードの変更を強制的に行わせるなどのインターフェースは用意されていません。そのため、これらの機能を実現したい場合は、JNDI（Java Naming and Directory Interface）API（Application Program Interface）などでLDAPの項目を編集するなどのアプリケーションを自作する必要があります。

Access Managerには、パスワード・ポリシー設定のためのコマンドが用意されていて、パスワードの文字長さ、パスワードの有効期限、パスワード入力失敗数によるパスワードの失効などのポリシーを設定できます。

・費用

認証プロキシーを使用する場合は、それなりの性能があるハードウェアを用意する必要があります。可用性や負荷分散を考えて、2台構成にする場合が多いため、やはり、そのハードウェア分の費用が必要になります。

3. おわりに

WebSphere Portalで、ポータル・サイト構成を考える際のポイントは、「できるだけシンプルな構成にすること」です。ユーザー・レジストリーを統一し、バックエンド連携もできるだけ簡単な方式を選択します。そのために既存のアプリケーションやユーザー・レジストリーに変更を加える必要が生じても、将来的な運用面や管理面を考えて、シンプルな構成への移行を検討しましょう。それでもユーザー・レジストリーが別になってしまうなどの特別な対応が必要な場合は、アプリケーションでの作り込みを考えるより前に、製品からユーティリティーとして提供されている機能をできるだけ活用することを考える必要があります。

本論文が、複数の環境を統合するポータル・サイトの構築において、最適な構成を選択するための一助になれば幸いです。

（ページ数および表記上の観点から、著者の了解を得て編集部にて手を入れてあります）

[参考文献]
[1] Gene Phifer, Gartner Group,‘ Portal ’: The Most Abused Term in IT ’ 2000
[2] ISE Technical Conference「WebSphere Portal Serverによるポータル・サイト構築」2001 http://www.ibm.com/jp/ise/service/techconf/2001/pdf/p02.pdf
[3]『ビジネスコミュニケーション』2001 Vol.38 特集「エンタープライズポータル最新動向」http://www.ntts.co.jp/ps/autonomy/images/bisicom1.pdf
[4] 富士通総研（FRI）「サイバービジネスの法則集」基礎知識「ポータルとは」http://www.fri.fujitsu.com/hypertext/fri/cyber/hotkey/portal/portal.html
[5] White Paper「WebSphere Application Server, V3.5 Security Overview」http://www-3.ibm.com/software/webservers/appserv/support.html