セキュリティー設計

当セッションの目的

- WASセキュリティー設計に必要な知識の習得
  - WASを使用したシステムにおいて、以下のセキュリティー対策毎の設計手法を理解する
    - ハードニング
    - 認証
    - 認可
    - 暗号化 / 署名
    - 監査

Agenda

1. はじめに

2. WASセキュリティー設計
   - 2-1　ハードニング
   - 2-2　認証
   - 2-3　認可
   - 2-4　暗号化 / 署名
   - 2-5　監査

まとめ・参考文献

1. はじめに

Webシステムにおける脅威と対策

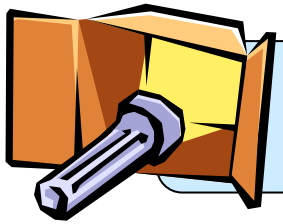
脅威	対策	WAS設計
サービス妨害 セキュリティー・ホール悪用	ハードニング	ゾーニング コンポーネントの制限 非rootユーザー稼動 IHSセキュリティー対策
なりすまし	認証	グローバル・セキュリティー 複数セキュリティー・ドメイン シングル・サインオン (SSO)
不正アクセス	認可	管理ロール 詳細な(Fine-grained)管理セキュリティー アプリケーション・セキュリティー
情報の盗聴 情報の改竄 不正コードの埋め込み	暗号化 / 署名	SSL 証明書
否認	監査	セキュリティー監査

トレードオフ:

- セキュリティー強化　vs.　ユーザーの利便性向上
- セキュリティー強化　vs.　パフォーマンス向上

セキュリティー関連情報

- セキュリティー・ホールは事前に全てを見つけられないので、日々の運用の中で、下記の対策を実施して下さい



インターネット・サイトの運営にあたり
セキュリティー対策は必須です

- 対策1：情報収集
  - 各製品に関連するセキュリティー関連の情報や、セキュリティー対策を施したFixが提供されているかを確認する

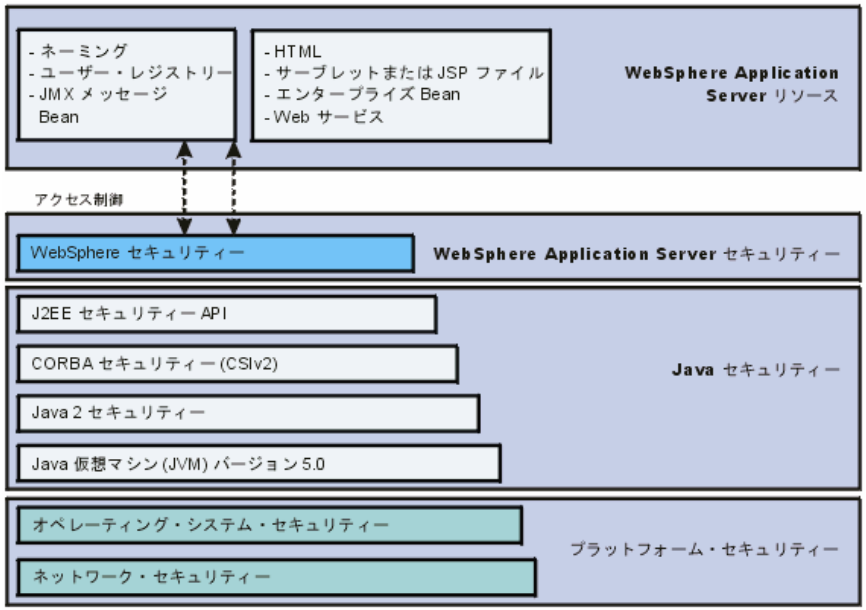
<重要セキュリティー情報サイト>
http://www.ibm.com/ibm/jp/security/info/

<WebSphere関連>
http://www.ibm.com/ibm/jp/security/info/WS.html

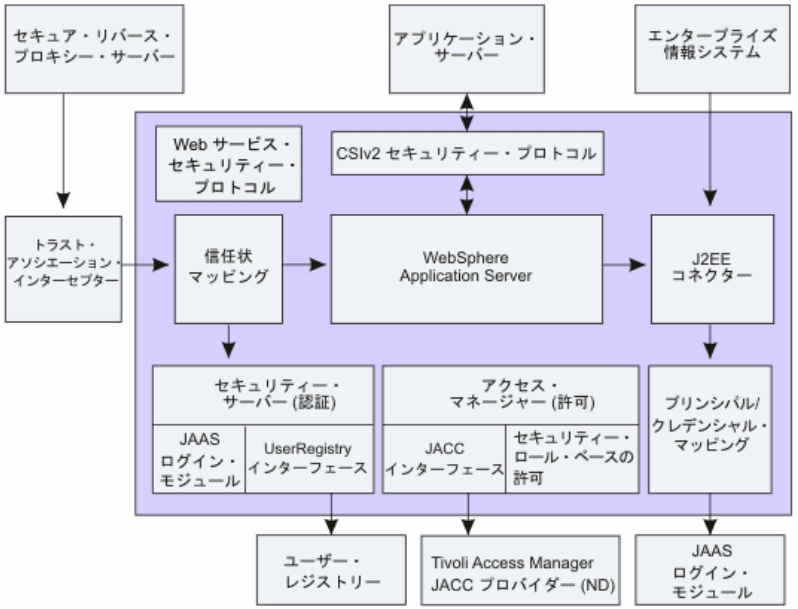
- 対策2：セキュリティー・Fixの適用
  - セキュリティー・ホールへの対応策、もしくはFixが提供された場合は速やかに適用する

WebSphereセキュリティーを支える技術

階層化セキュリティー・アーキテクチャー



オープン・アーキテクチャー・パラダイム



Java 2 セキュリティーは、「セキュリティー設計-参考」のP.80をご参照下さい。
JACCは、「セキュリティー設計-参考」のP.81をご参照下さい。
JAASは、「セキュリティー設計-参考」のP.82をご参照下さい。

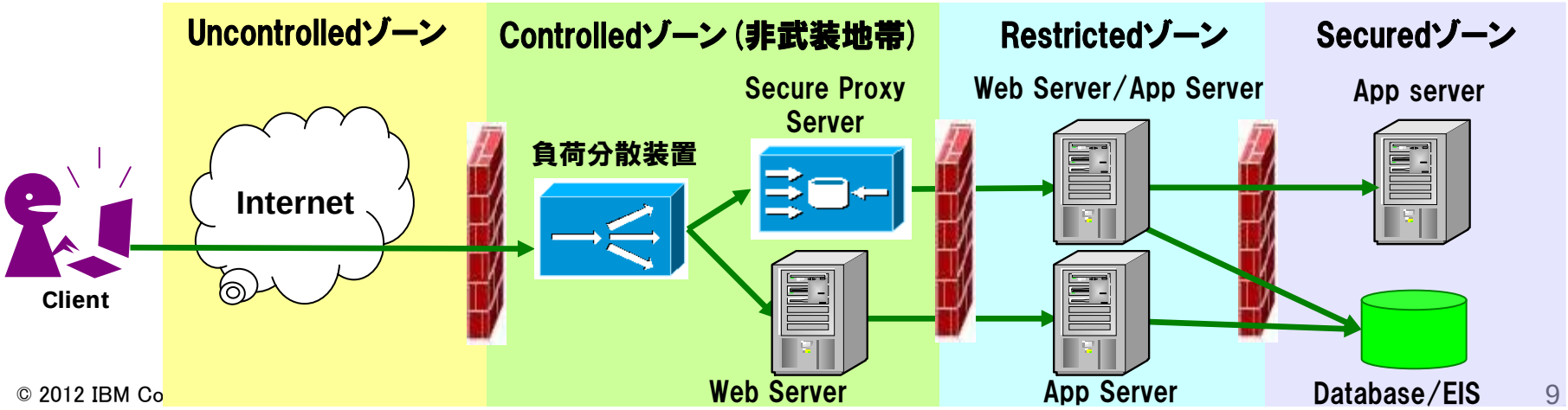
2. WASセキュリティー設計

2-1 ハードニング
2-2 認証
2-3 認可
2-4 暗号化 / 署名
2-5 監査

ハードニングとは、必要な機能以外は停止、遮断すること

ゾーニング設計

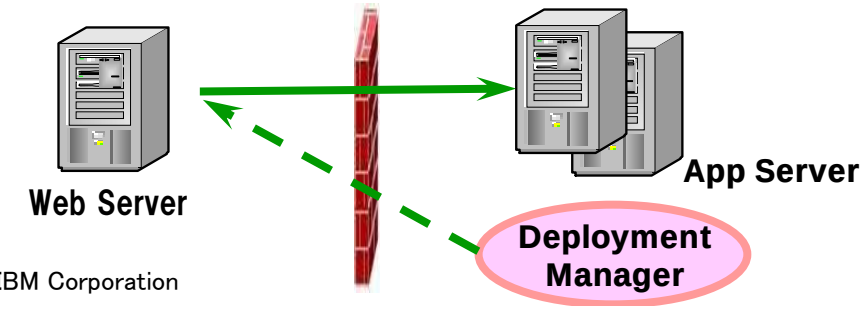
- ゾーニングとは
  - サーバーやネットワーク機器などのノードを、セキュリティー・レベルに応じて分割する
- ゾーンの分離
  - Load Balancer、Webサーバー、Proxy Server (Secure Proxy Server)のDMZへの配置
    - 外部からの不正なアクセスを遮断する
    - 内部ネットワークのマシンを、直接的な攻撃対象とさせない
    - 内部ネットワークのIPアドレスなどの情報を隠蔽させる
  - SSL通信の解除
    - 前段で解除した方がパフォーマンスが良い
    - 後段で解除した方がセキュリティーが向上する



ファイアウォール設定

- 設計指針
  - 必要なポート以外はオープンしない

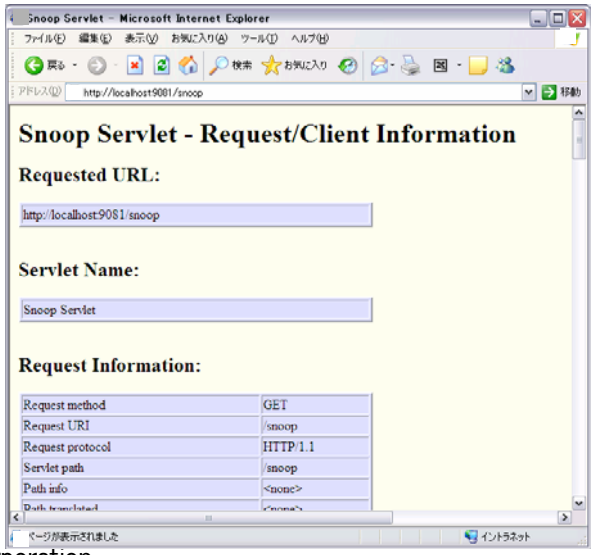
ホスト	デフォルト・ポート(TCP)	向き	ホスト	デフォルト・ポート(TCP)	備考
Web Server	*	→	App Server	9080	アプリケーション・サーバーのリクエスト受付ポート(HTTP)
Web Server	*	→	App Server	9443	アプリケーション・サーバーのリクエスト受付ポート(SSL)
Web Server (IHS only)	8008	←	Deployment Manager (管理サーバー)	*	IHS管理サーバーのポート。リモートWebサーバーがIHSの場合、Webサーバー管理のため。
Web Server	80	←	Deployment Manager (管理サーバー)	*	Webサーバーのlistenポート。管理サーバーが、Webサーバーの稼動状況確認のため。



WASやIHSのListenポートを変更している場合には、適宜変更する

コンポーネントの制限

- アプリケーション
  - セキュリティー上好ましくないデフォルト・アプリケーションを削除する
    - Snoopサーブレット、HitCountアプリケーション等
  - セキュリティー上好ましくないサンプル・アプリケーションを導入しない
    - キャッシュ・モニター・アプリケーション等
- ポート番号
  - デフォルトのポート番号は、限定的で特定しやすいため、変更する
    - 変更例：WC_defaulthost　　9080 →19080



ポート名	ポート
BOOTSTRAP_ADDRESS	9810
SOAP_CONNECTOR_ADDRESS	8880
ORB_LISTENER_ADDRESS	9101
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9404
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9405
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9406
WC_adminhost	9061
WC_defaulthost	9080
DCS_UNICAST_ADDRESS	9354
WC_adminhost_secure	9044
WC_defaulthost_secure	9443
SIP_DEFAULTHOST	5060
SIP_DEFAULTHOST_SECURE	5061
IPC_CONNECTOR_ADDRESS	9633
SIB_ENDPOINT_ADDRESS	7276
SIB_ENDPOINT_SECURE_ADDRESS	7286
SIB_MQ_ENDPOINT_ADDRESS	5558
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5578

非rootユーザー稼動

- 目的
  - クライアントから直接アクセスされるWebサーバー、Webアプリケーション・サーバーをrootで稼動させた場合、セキュリティー低下につながる
  - 侵入されてしまった場合の被害を最小限にする
- IHS / WASともに非rootでの稼動が可能
  - IHSは、1024番以降のポート番号とする必要がある
    - AIX V6.1の新機能である拡張RBAC(Role Based Access Control)機能を使用すると、80番ポートを使用してIHSを非rootで稼動させることが可能
  - IHSをWASとは異なる非rootユーザーで稼動し、かつ管理コンソールから起動/停止/構成管理を行う場合は、IHS管理サーバー経由とする必要がある
  - 非rootユーザーによる製品インストール、CIM、Fixpackの適用も可能
  - ただし、IHSはもともとrootプロセスと非rootプロセスの組み合わせで動作するようになっているため、非rootユーザーでの稼働の意味は少ない

<WASの場合>
非rootユーザーで起動・停止を行うJVMを含むプロファイル・ディレクトリー、プロファイル・ログ・ディレクトリーのオーナーを変更
<IHSの場合>
IHSインストール・ディレクトリーのオーナーを変更
httpd.confのListenディレクティブで指定するポート番号を1024以降に変更
<IHS管理サーバーの場合>
IHSインストール・ディレクトリーのオーナーを変更
admin.confのUser、Groupディレクティブに非rootユーザーを指定
非rootユーザーにて、htpasswdコマンドを使用し、IHS管理サーバー内で使用するユーザーを作成
httpd.confのListenディレクティブで指定するポート番号を1024以降に変更する

```
# su – wasadmin
$ <WAS_PROFILE_ROOT>/bin/startNode.sh
wasadmin  708764       1   1 17:09:17  pts/0  0:29
/usr/IBM/WebSphere/AppServer/java/bin/java <中略>
nodeagent


#su – ihsadmin
$ /usr/IBM/HTTPServer/bin/apachectl –k start
ihsadmin  540884       1   0 17:21:20      -  0:00
/usr/IBM/HTTPServer/bin/httpd -d
/usr/IBM/HTTPServer -k start
```

IHSのセキュリティー対策　～バージョン情報の隠蔽

- 目的
  - HTTPレスポンス・ヘッダーやエラー画面のフッターにIHSのバージョン情報が表示され、悪意のあるユーザーからセキュリティー・ホールを狙われてしまう可能性がある
- 設定方法
  - httpd.confの確認
  - WAS V7以降は
    デフォルトで設定されているので変更しないことを推奨

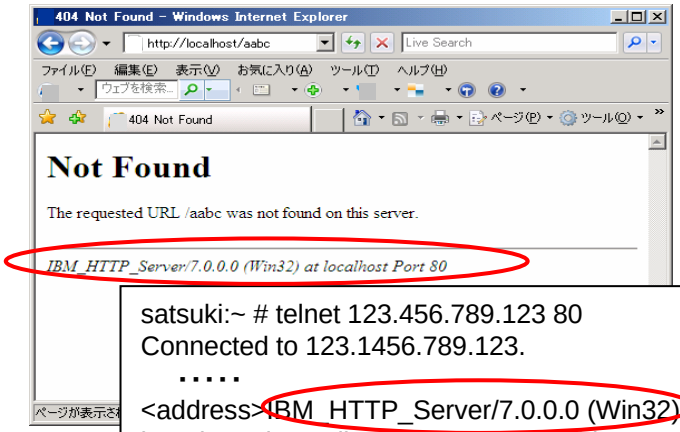
```
ServerTokens ProductOnly
ServerSignature Off
```

HTTPレスポンス・ヘッダーに関する設定で「Prod」でも「ProductOnly」でもOK

サーバーが生成するドキュメント（エラー画面など）のフッターに関する設定

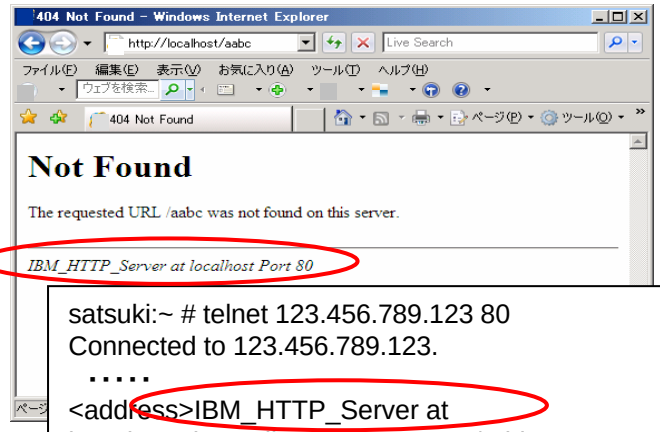
- バージョン差異による考慮事項
  - IHS V6.0以降（Apache2.0.44以降）　ServerTokensを「Prod」に設定
  - それより前のバージョンの場合　ServerTokensとServerSignatureを設定

＜上記の設定になっていない場合＞




```
satsuki:~ # telnet 123.456.789.123 80
Connected to 123.1456.789.123.
    .....
<address>IBM_HTTP_Server/7.0.0.0 (Win32) at
hogehoge.japan.ibm.com Port 80</address>
</body></html>
Connection closed by foreign host.
```

＜上記の設定の場合＞




```
satsuki:~ # telnet 123.456.789.123 80
Connected to 123.456.789.123.
    .....
<address>IBM_HTTP_Server at
hogehoge.japan.ibm.com Port 80</address>
</body></html>
Connection closed by foreign host.
```

IHSのセキュリティー対策　～TRACEメソッドの無効化

- 目的
  - クライアントが送信したリクエスト・メッセージをそのまま返す機能である
  - TRACEメソッドが有効になっていると、悪意のあるユーザーが、別ユーザーのブラウザーにてこのTRACEメソッドを発行するように仕向けてそのレスポンスの中のパスワードを奪うという Cross Site Tracing と呼ばれる攻撃を受ける可能性がある
- 設定方法
  - httpd.confの確認

```
TraceEnable Off
```

  - デフォルトではOFFなので変更しないことを推奨
- バージョン差異による考慮事項
  - IHS V7.0（Apache2.0.55以降）　上記設定で無効化できる
  - それより前のバージョンの場合　以下のように定義する必要あり

```
LoadModule rewrite_module modules/mod_rewrite.so
RewriteEngine On
RewriteCond %{REQUEST_METHOD} ^TRACE
RewriteRule .* - [F]
```

＜上記の設定になっていない場合＞

```
satsuki:~ # telnet 123.456.789.123 80
Connected to 123.456.789.123.
OPTIONS / HTTP/1.0
  .....
Server: IBM_HTTP_Server
Allow: GET,HEAD,POST,OPTIONS,TRACE
```

＜上記の設定の場合＞

```
satsuki:~ # telnet 123.456.789.123 80
Connected to 123.456.789.123.
OPTIONS / HTTP/1.0
  .....
Server: IBM_HTTP_Server
Allow: GET,HEAD,POST,OPTIONS
```

IHSのセキュリティ対策　～リクエストの制限

- 目的
  - クライアントからの異常なリクエストを制御し、DoS攻撃の危険性を低減させる
- 設定方法
  - httpd.confの確認
  - IHSにおいて制限可能な項目
    - LimitRequestLine
      - リクエスト・ラインの長さ（URLの長さ）の制限
    - LimitRequestFields
      - リクエストのヘッダの数の制限
    - LimitRequestFieldSize
      - リクエストのヘッダの長さの制限
    - LimitRequestBody
      - POSTリクエストのBodyのサイズ制限
    - LimitXMLRequestBody
      - XML形式のリクエストのBodyのサイズ制限

	デフォルト値
LimitRequestLine	LimitRequestLine 8190
LimitRequestFields	LimitRequestFields 100
LimitRequestFieldSize	LimitRequestFieldSize 8190
LimitRequestBody	LimitRequestBody 0
LimitXMLRequestBody	LimitXMLRequestBody 1000000

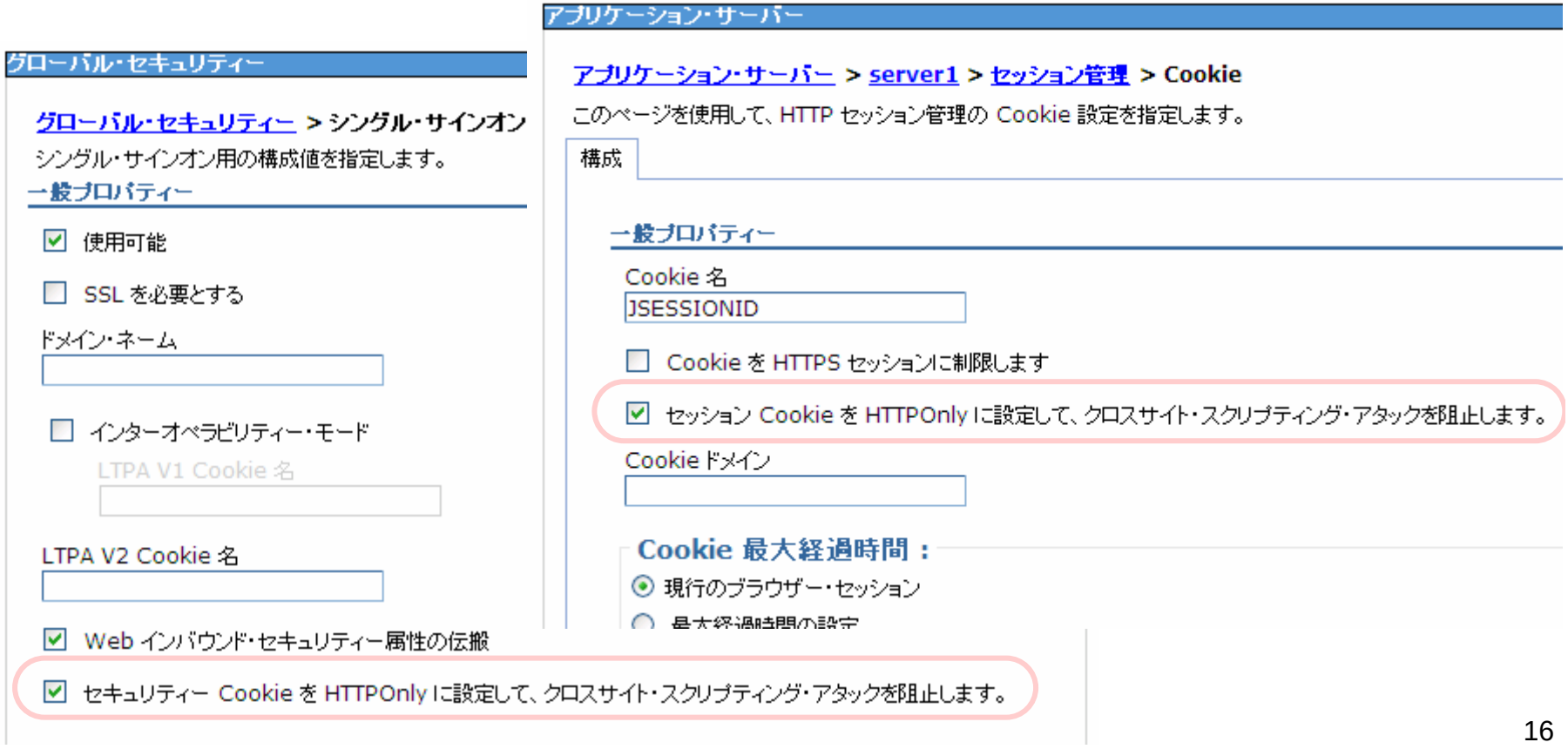
実際にアプリケーションで使用する量＋αを設定する

ハードニングの強化（1）

Update v8

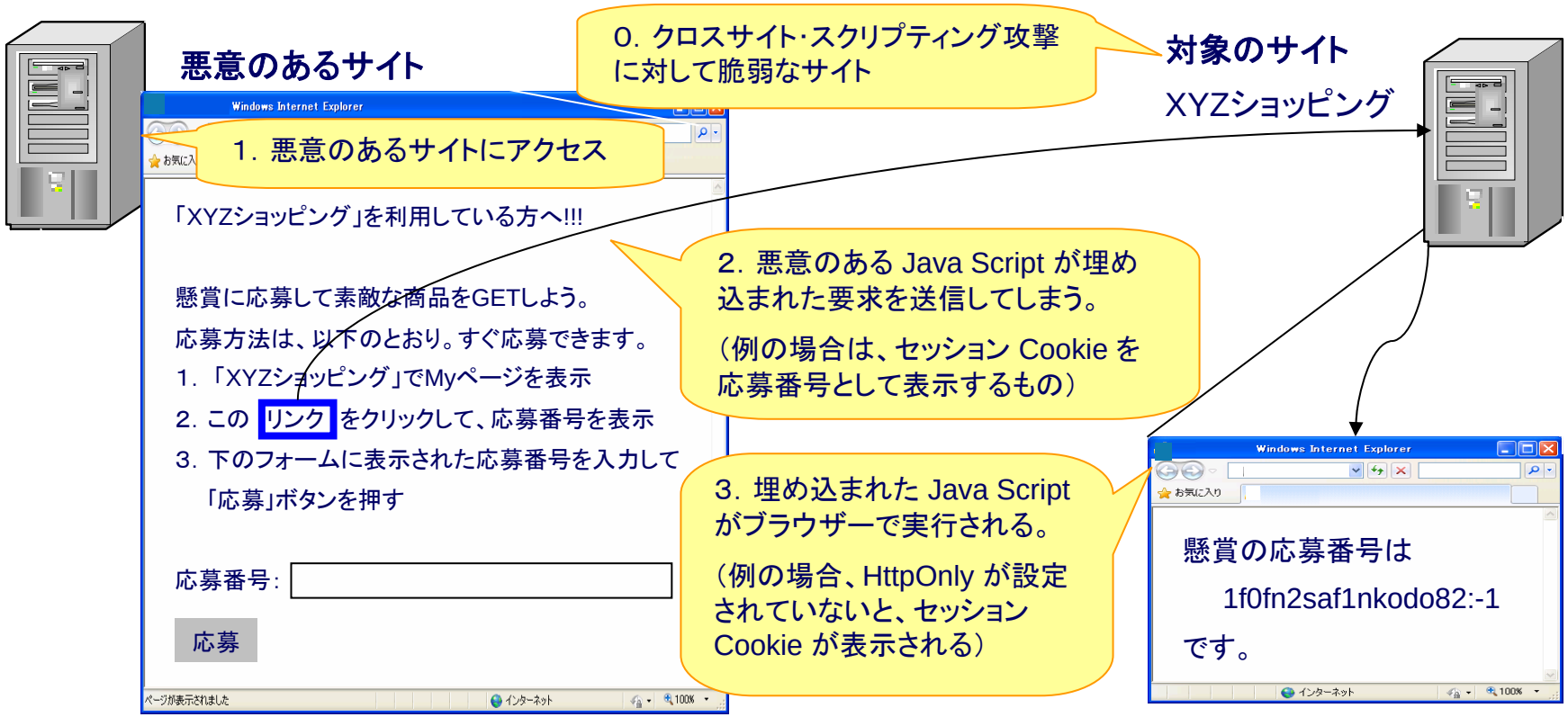
- クロスサイト・スクリプティング攻撃対策として、Cookie の HttpOnly 設定がデフォルトで on
  - HTTP セッション管理用の Cookie
  - LTPA 関連の Cookie

◆HTTPOnlyにすることで、JavaScriptがCookieにアクセスできなくなるので、XSS攻撃の代表であるCookie盗難の危険を軽減



【参考】クロスサイト・スクリプティング攻撃

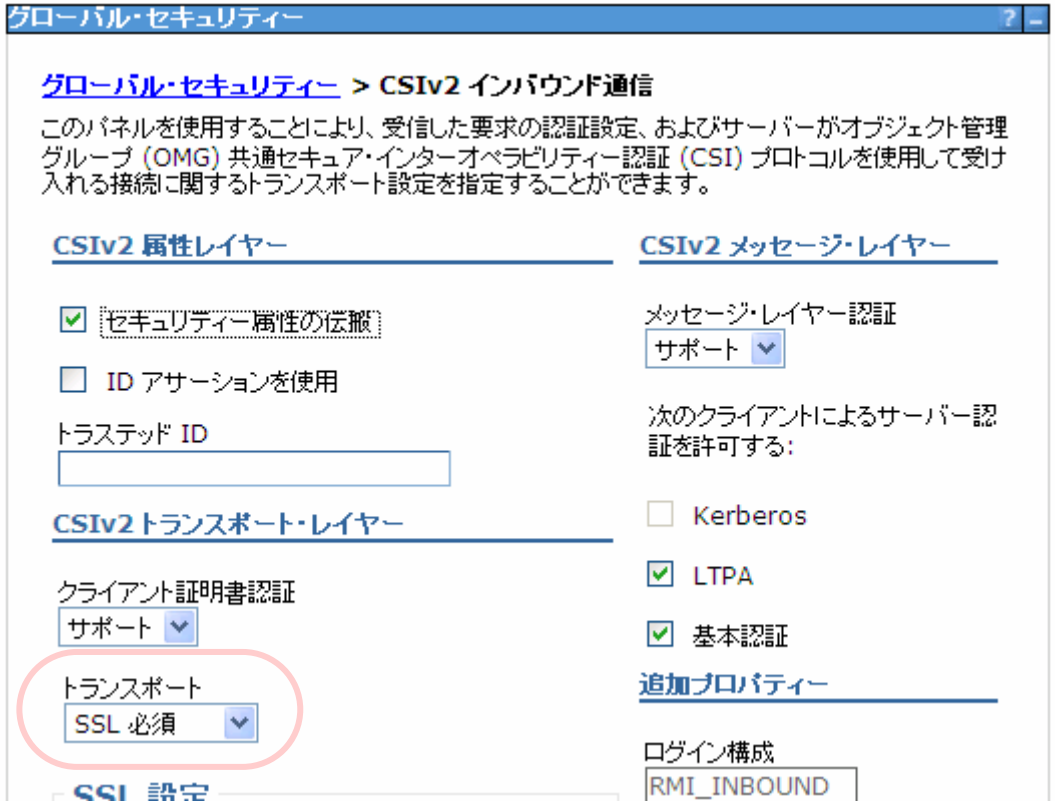
- クロスサイト・スクリプティング攻撃
  - ユーザーのブラウザーを経由して、悪意のあるサイトから対象のサイトにスクリプトが送り込まれ、ユーザーまたは対象のサイトに被害が発生する
  - HttpOnly 設定をしていない場合や、HttpOnly 設定を認識しないブラザーを使用していると、容易に Cookie が読み取られる

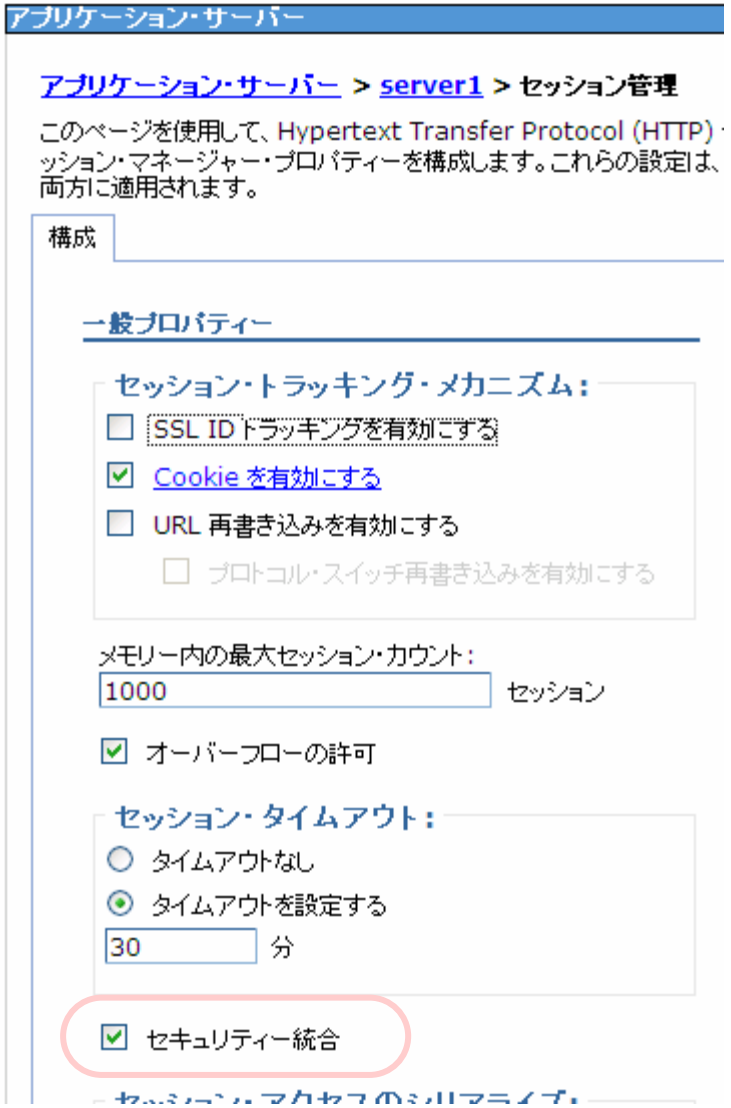


ハードニングの強化（2）

Update v8

- HTTP セッションのハードニング強化
  - セッション管理の「セキュリティー統合」がデフォルトで on
  - グローバル・セキュリティを有効にする必要あり
- RMI/IIOP 通信のハードニング強化
  - CSIv2 インバウインド&アウトバウンド通信がデフォルトで SSL 必須
  - WAS V7以前は、最初に平文通信を行い、不可能なときにSSL通信に切り替える仕組みだったが、V8ではデフォルトでSSL通信を行う

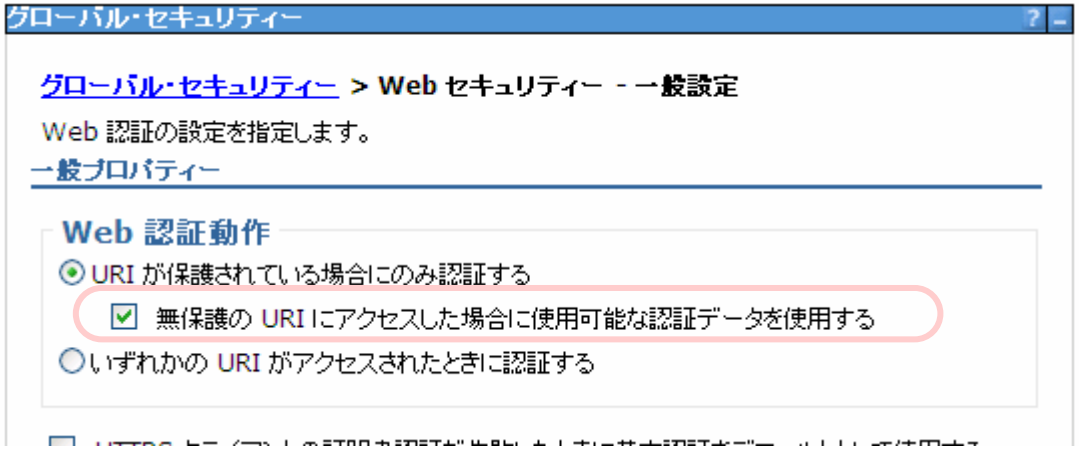




ハードニングの強化（3）

Update v8

- 無保護リソースのアクセス時も認証データがデフォルトで利用可能に
  - SSL通信を行う



- その他のハードニング強化
  - デフォルトの SSL 鍵サイズの強化
    - 1024 ビット → 2048 ビット
  - 使用可能なSSL鍵サイズの拡張
    - 8192 ビットを追加

2. WASセキュリティー設計

2-1 ハードニング
2-2 認証
2-3 認可
2-4 暗号化 / 署名
2-5 監査

認証とは、利用者が正当なユーザーであることを確認すること

認証

- アクセス管理　= 認証 ＋　認可

- 認証(authentication)
  - 利用者の本人性を特定すること
    - 知っていること　　ユーザーIDやパスワードなど
    - 持っていること　　IDカード、電子証明書など
    - 生体個別の特徴を有すること　　指紋など（WASでは直接扱いません）

- WASで認証を行うためには、グローバル・セキュリティーの設定が必要

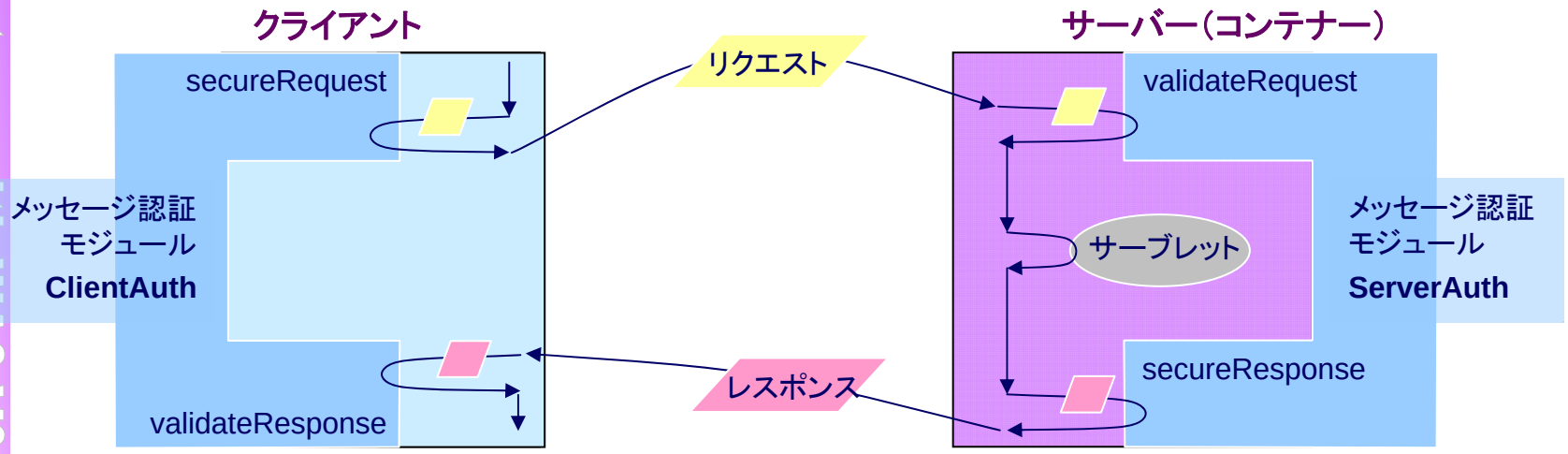
WASがサポートする認証メカニズム

- 基本認証
- フォーム認証
- クライアント証明書認証
- Kerberos 認証
- 内部認証
  - JAAS ログイン・モジュールを使用したプログラマティック・ログイン
- JASPI（Java 認証 SPI）認証 *New v8*
  - JASPI プロバイダーによる認証
- LTPA （Lightweight Third-Party Authentication）
- SPNEGO Web 認証
- トラスト・アソシエーション

JASPI（Java 認証 SPI）認証

New v8

- メッセージ認証（Message Authentication）の仕組みで、JSR-196「Java Authentication SPI for Containers」として規定
  - 認証プロバイダーがメッセージ認証モジュールを提供
  - コンテナーがメッセージ認証モジュールを呼び出すことで認証などを実施
    - サーバー側： リクエストを検証（認証）し、レスポンスをセキュア化
    - クライアント側： リクエストをセキュア化し、レスポンスを検証
  - コンテナーへの外部認証機能の組み込みが容易に
- 各々のメッセージ処理形態ごとにプロファイルを定義
  - Servlet Container, SOAP, JMS, RMI/IIOP Portable Interceptor, LoginModule Bridge
- WAS V8 は、Servlet Container プロファイル（HttpServletプロファイル）のみをサポート



グローバル・セキュリティー

- WAS自体および連携する他のコンポーネントを含むシステムの一部を保護された状態(認証、認可、暗号化の実施など)にする機能
  - WAS V6.1から管理セキュリティーとアプリケーション・セキュリティーが分離
  - 管理セキュリティーのみの設定が可能
  - ジョブ・マネージャー、管理エージェントに対しても設定が可能

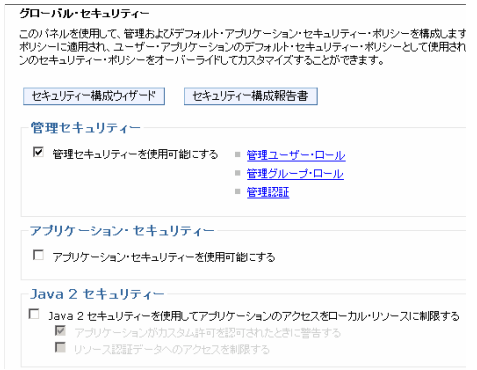
- 管理セキュリティー
  - WASシステムに対する管理全般に対して、認証・認可を設定する
    - (例)管理コンソール、管理コマンド等
  - プロファイル作成時のデフォルトはON、プロファイル作成後でも変更可能

- アプリケーション・セキュリティー
  - WAS上のアプリケーション・リソースに対して、認証・認可を設定する
    - (例)アプリケーション利用ユーザーに認証を求める
  - 前提:管理セキュリティーを有効にする

グローバル・セキュリティーの設定方法は、「セキュリティー設計-参考-」のP.84をご参照下さい。

管理セキュリティー

- WASシステムの管理をセキュアーにする機能
  - WASのシステムの管理を行うユーザーを制限したい場合に設定する



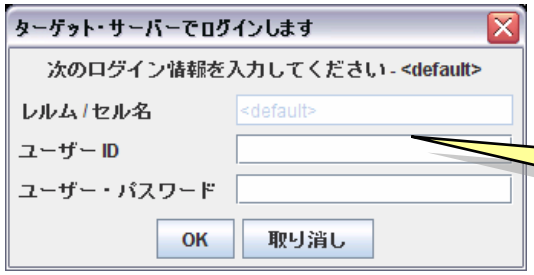
- 特徴
  - WAS管理のみセキュアーにできる
  - ウィザードにより簡単に構成できる
  - WAS導入時に簡単に構成できる
  - フェデレーテッド（統合）リポジトリーの使用により、簡単にユーザ・レジストリーを構築できる

- 管理セキュリティーをONに設定すると、
  - LTPAを使用して認証を行う
  - コンポーネント間の通信が自動的にSSL化される
    - WAS間(DM,NA,APP)の通信、管理ツールからのアクセスがSSLとなる
    - Plug-in（IHS等）、WAS間の通信は、自動的にSSL通信とはならない
  - 管理コンソールログイン時、wsadminコマンド実行時、WAS停止時にユーザーIDとパスワードが必要

wsadminコマンド / WAS停止コマンドの実行

- 管理セキュリティーを設定すると、wsadminコマンド実行時、WAS停止コマンド実行時にユーザー認証が必要となる

```
stopNode –username xxxx –password xxxx
```



ユーザーIDとパスワードを指定しないと入力用プロンプトが表示される

- wsadminコマンド / WAS停止コマンド実行の自動化
  - soap.client.propsファイルを編集

```
com.ibm.SOAP.securityEnabled=true
com.ibm.SOAP.loginUserid=xxxx
com.ibm.SOAP.loginPassword=xxxx
```

    - 実行時プロンプトの制御
      - 自動実行時にユーザー入力用のプロンプトを表示させたくない場合に設定する

```
com.ibm.SOAP.loginSource=none
```

    - パスワードの暗号化
      - パスワードを設定後、PropFilePasswordEncoderコマンドを実行する

```
PropFilePasswordEncoder <WAS_PROFILE_ROOT>/properties/soap.client.props
com.ibm.SOAP.loginPassword
```

パスワード変更コマンド

New v8

- 新しいwsadminコマンド “changeMyPassword”が追加
  - WASにログインしたユーザーが自身のパスワードを変更可能
  - 管理コンソール上の操作は非サポート
  - フェデレーテッド(統合)リポジトリー用の書き込みアダプターを持つリポジトリーに対してのみ使用可能
  - 認証キャッシュを使用していると、古いパスワードでログインできる可能性がある
    - キャッシュをクリアにすれば、キャッシュのタイムアウトを待たずに、新しいパスワードを有効にできる
- コマンド例(Jython形式):
  - $AdminTask changeMyPassword (‘[-oldPassword password -newPassword passw0rd -confirmNewPassword passw0rd]’)

```
wsadmin>AdminTask.changeMyPassword ('[-oldPassword password -newPassword passw0r
d -confirmNewPassword passw0rd]')
'CWWIM5099I Command completed successfully.'
wsadmin>
```

アプリケーション・セキュリティー

- WAS上のアプリケーション・リソースをセキュアーにする機能
  - アプリケーション・リソースへのアクセスを制限したい場合に設定する
- Web サーバーへのユーザー認証
  - 基本認証（Basic）
    - WebサーバーがWebクライアントに認証を要求し、Web クライアントはHTTP ヘッダーでユーザー ID とパスワードを渡す
  - フォーム認証（Form）
    - ログイン・ページにHTMLフォームを使用する認証です
  - 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、ユーザーの認証を行う
    - HTTPSプロトコルを使用する

・基本認証

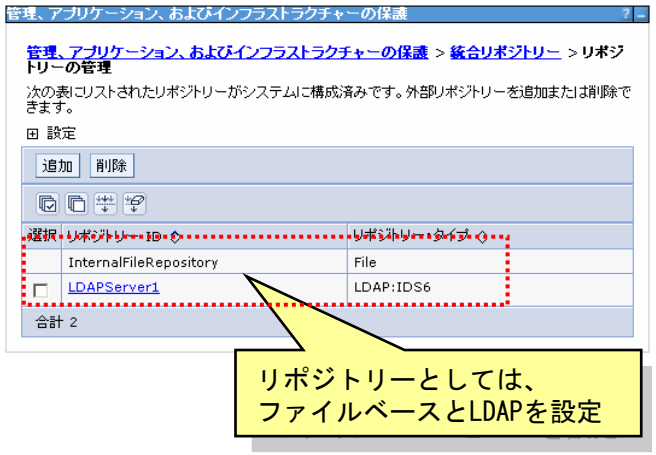
・フォーム認証

認証リポジトリー（ユーザー・アカウント・リポジトリー ）

- 以下の4種類をサポート
  - ◆フェデレーテッド（統合）リポジトリー
    - 複数のレジストリーを論理的な1つのレジストリーとして利用（ WAS V6.1からの機能 ）
    - 以下の4つレジストリーを組み合わせることが可能
      - ファイル・ベースの組み込みリポジトリー
      - LDAP
      - カスタム･レジストリー
      - ファイル･レジストリー

Update v8

    - デフォルトは、ファイル・ベースの組み込みリポジトリーで構成
      - fileRegistry.xmlファイルにて管理する
      - ユーザー/グループ管理は、管理コンソールから実施



  - ◆スタンドアロンLDAPレジストリー
    - LDAPをユーザー･レジストリーとして利用
  - ◆スタンドアロン・カスタム・レジストリー
    - データベースやテキスト･ファイルなどをレジストリーとして利用する際に選択
    - UserRegistry Javaインターフェースを使用し、ユーザーが独自に実装する必要がある
  - ◆ローカル・オペレーティング・システム・レジストリー
    - WASが導入されているOSのユーザー登録情報をレジストリーとして利用
    - シングル・サーバー構成でのみ選択可能

注意

【参考】認証リポジトリーの選択指針

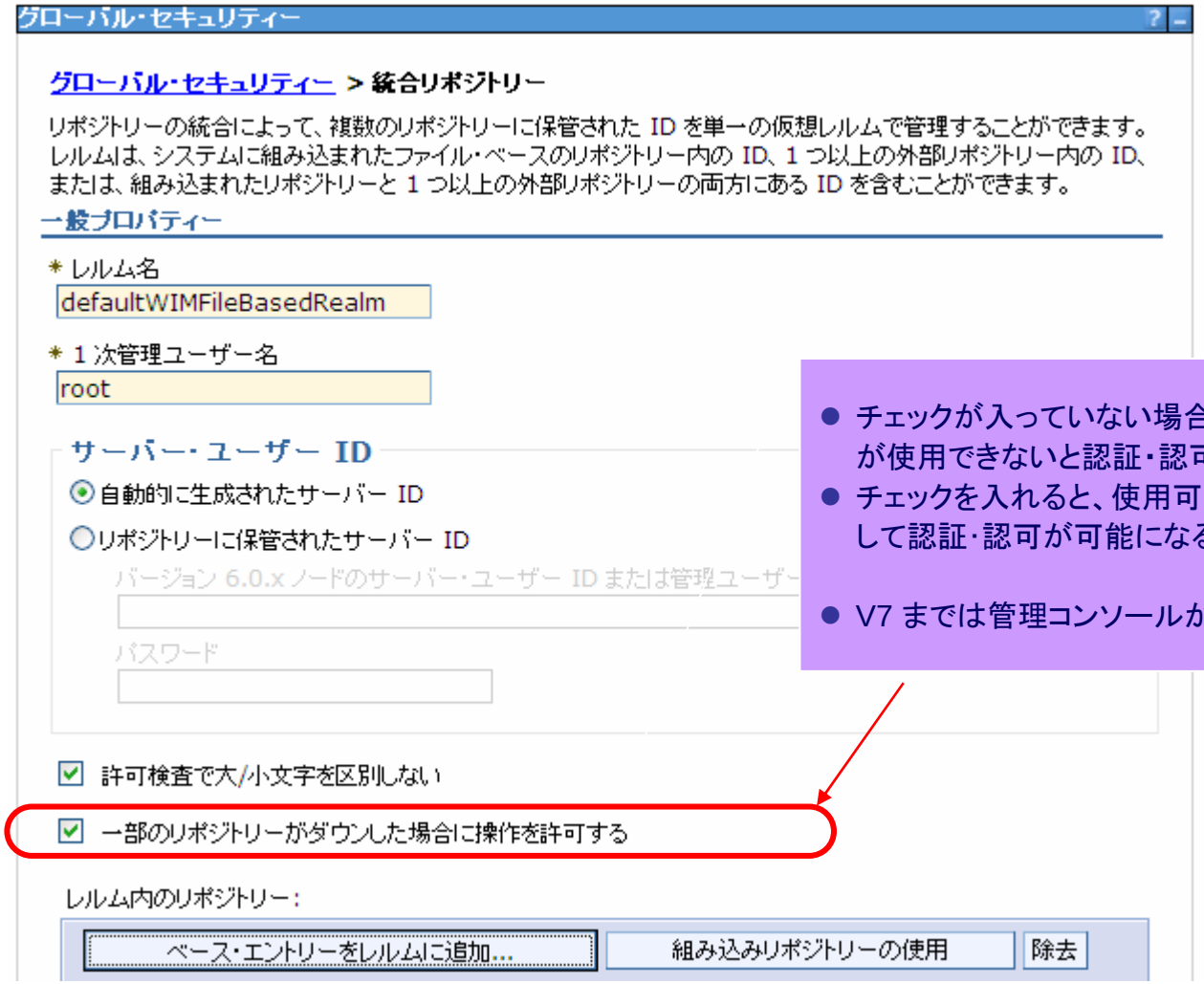
	①フェデレーテッド（統合）リポジトリー	②スタンドアロンLDAP	③ローカル・オペレーティング・システム	④スタンドアロン・カスタム
レジストリー	ファイルベースの組み込みリポジトリ、LDAP、ファイル、カスタム	スタンドアロンLDAP	ローカル・オペレーティング・システム	スタンドアロン・カスタム
同時複数リポジトリーサポート	Yes	No	No	No
リポジトリーRead/Writeサポート	read/write	read	read	read
レルム名の変更	管理コンソールから変更可能	管理コンソールから変更可能	変更できない	変更できない
事例	・WAS管理ユーザーを既存のLDAP環境に登録できない場合に、フェデレーテッド（統合）リポジトリーを使用しWAS管理ユーザーをファイル、ユーザー情報をLDAPにて管理する ・将来的に、現在使用しているリポジトリー以外の方式を選択する可能性がある場合	・LDAPでしかユーザー情報を管理しない場合 ・既存構成から変更したくない場合	・WAS Base構成で、WAS管理ユーザーをOSで管理したい場合 ・ WAS Base構成で、WAS管理セキュリティーのみを構成する場合	・①/②/③では対応できない場合 ・UserRegistoryのモジュールが提供されているパッケージを使用する

1セル内で複数の認証リポジトリーの設定が必要な場合には、複数セキュリティー・ドメインにて対応する

フェデレーテッド(統合)リポジトリーの改良・変更

New v8

- フェデレーテッド(統合)リポジトリーにより複数レジストリーを統合している構成での、レジストリー障害発生時の動作設定が簡単に

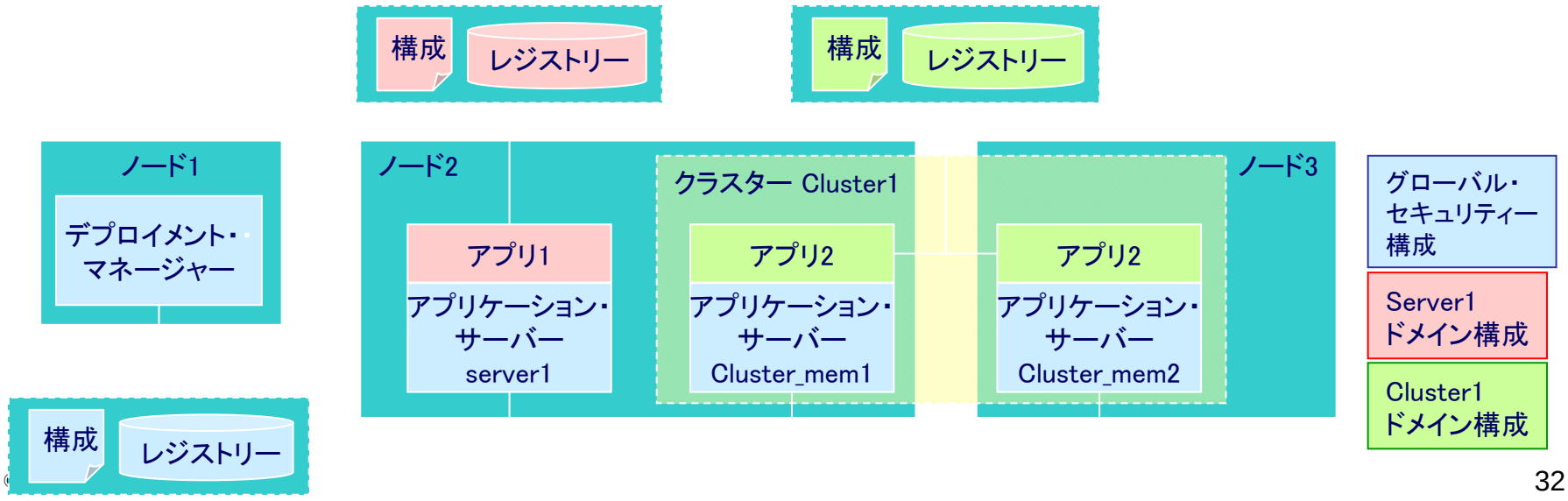


- チェックが入っていない場合、1つでもレジストリーが使用できないと認証・認可が一切行えなくなる
- チェックを入れると、使用可能なレジストリーを使用して認証・認可が可能になる
- V7 までは管理コンソールから設定できなかった

複数セキュリティー・ドメイン

- WAS V6.1での課題
  - セルで設定可能なセキュリティー構成は1つのみ
  - 異なる認証リポジトリーを使用する場合は、セルを分割する
- WAS V7.0以降
  - セキュリティー・ドメインを定義することで、グローバル・セキュリティー構成を上書きすることが可能
  - グローバル・セキュリティー構成は、管理セキュリティーとデフォルトのセキュリティー構成となる
    - 例1： 管理セキュリティーではフェデレーテッド（統合）リポジトリーを、アプリケーション・セキュリティーでは LDAPレジストリーを使用する
  - セキュリティー・ドメインは複数定義可能
    - 例2： server1のアプリとCluster1のアプリが異なるレジストリーを使用する
- V8からセキュリティー・ドメインに対してフェデレーテッド（統合）リポジトリーを設定可能に
  - グローバル統合リポジトリー： グローバル・セキュリティー構成の統合リポジトリー

New v8



セキュリティー・ドメイン構成ファイル

- セキュリティー・ドメインの構成ファイル（手動編集不可）
  - security-domain.xml ファイル
    - セキュリティー・ドメインに構成されている属性リスト（ユーザー・レジストリー、ログイン構成等）
  - security-domain-map.xml ファイル
    - セキュリティー・ドメインを使用する有効範囲を含むファイル（サーバー、クラスター等）
- セキュリティー・ドメイン関連ファイル
  - 全てのセキュリティー・ドメインで、$SecurityDomainNameディレクトリーが作成
    - <WAS_PROFILE_ROOT>/config/waspolicies/default/securitydomains/$SecurityDomainName
- 上書き可能なセキュリティー属性一覧

グローバル・セキュリティー	セキュリティー・ドメインでオーバーライドが可能な項目
アプリケーション・セキュリティー	○
Java2セキュリティー	○
ユーザー・レルム(認証リポジトリー)	○
トラスト・アソシエーション(TAI)	○
SPNEGO Web認証設定	○
RMI/IIOPセキュリティー	○
JAAS	○
JASPI	○
認証メカニズム属性	○
許可プロバイダー	○
フェデレーテッド(統合)リポジトリー	○
カスタム・プロパティー	○
シングル・サインオン	
SSL	
セキュリティー監査	
LTPA認証メカニズム	
Kerberos認証メカニズム	

New v8

New v8

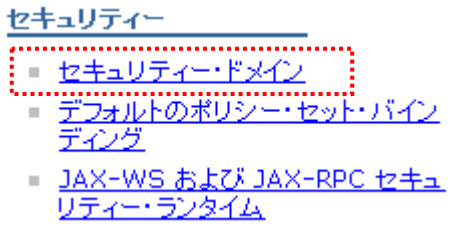
【参考】サーバー単位でのセキュリティー設定

注意

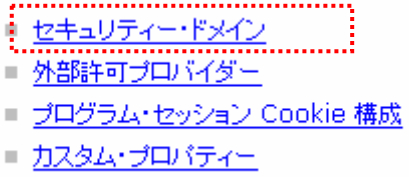
- WAS V7.0以降、サーバー・レベルでセキュリティー設定は非推奨です。サーバー・レベルで設定する場合は、複数セキュリティー・ドメインを使用して下さい。
- 背景
  - WAS V6.1では、サーバー単位でのセキュリティー設定が可能
  - WAS V7.0以降は、そもそもサーバー単位を選択できない

・WASV6.1での設定

・WASV7.0での設定



・WASV8.0での設定

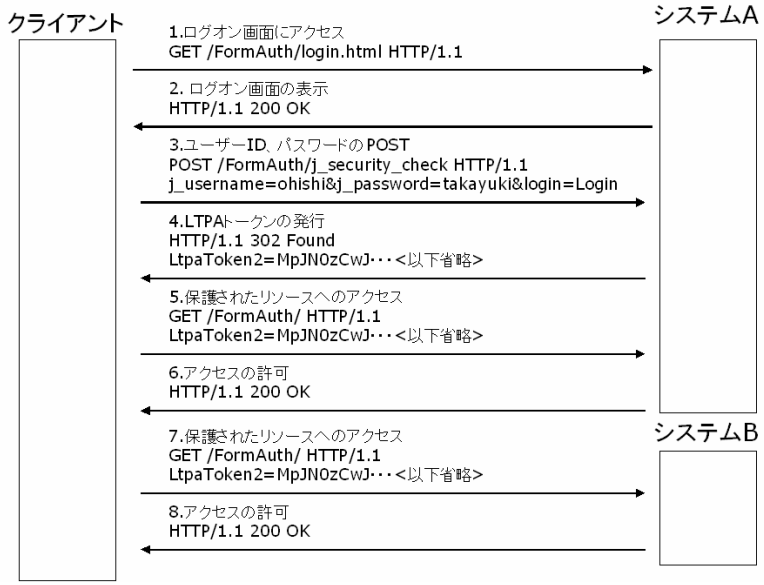


シングル・サインオン (SSO)

- 1回のログインで複数のシステムを利用可能にする認証のこと
  - 複数のシステムを利用する度に認証を実施する必要がなくなります

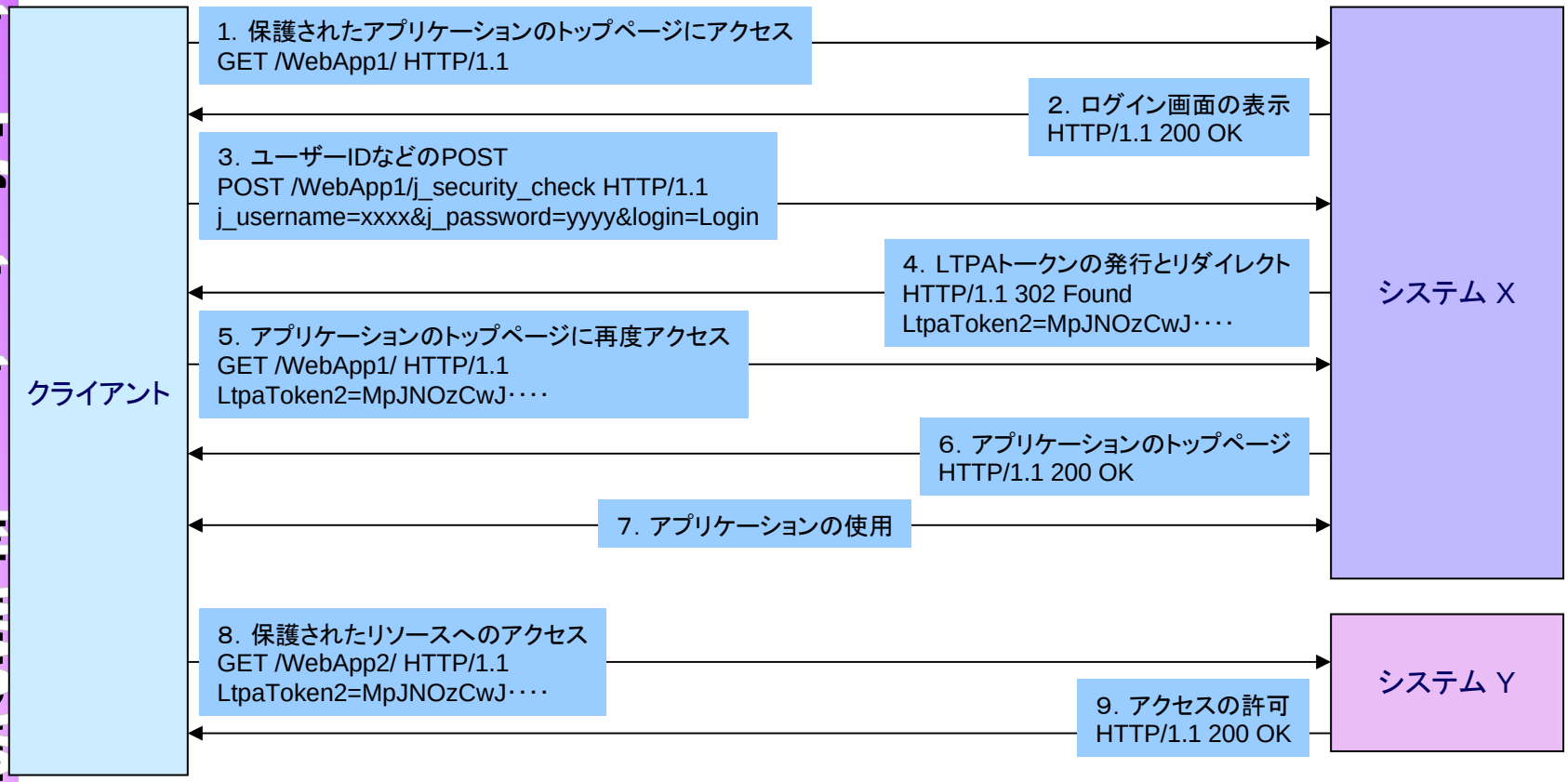
- WAS V8.0がサポートしている認証方法
  - LTPA認証
    - SSOを行うシステム間でLTPA鍵を共有する
    - 最初に認証を受けたサーバーがLTPAトークンを発行する
    - 2回目からのクライアントからのアクセスは、LTPAトークンを提示することで認証が不要となる
  - Kerberos認証
  - SPNEGO Web認証
  - トラスト・アソシエーション（TAM連携）

・LTPA認証の処理フロー



LTPA（Lightweight Third-Party Authentication）

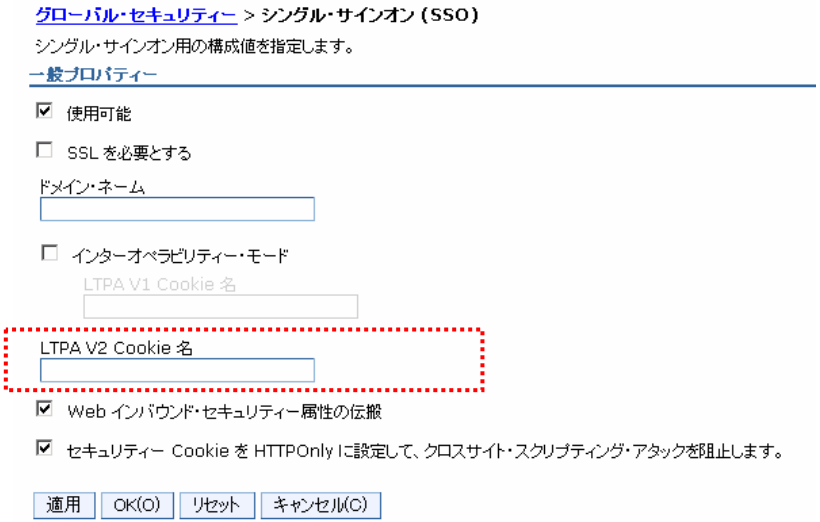
- IBM製品間で使用可能な認証技術
- LTPAトークンにより認証情報を伝播可能
  - セルや製品を跨いで伝播させる場合は、LTPA 鍵の交換が必要



LTPAの改良・変更

New v8

- LTPA 用 Cookie 名称のカスタマイズが簡単に
  - 「グローバル・セキュリティー > 「WebおよびSIPセキュリティー」>「シングル・サインオン (SSO) 」で LTPA 用 Cookie 名が指定可能
  - Cookie 名をカスタマイズするとSSOドメイン間の認証を論理的に分離でき、カスタマイズした認証を特定の環境で使用可能
    - 既存の別サーバーとの連携が簡単
  - 連携先のサーバーのCookie 名も要変更
  - 混合セル環境では使用不可
- LTPA 鍵のインポート/エクスポートが管理スクリプトで実行可能に
  - AdminTask の importLTPAKeys / exportLTPAKeys



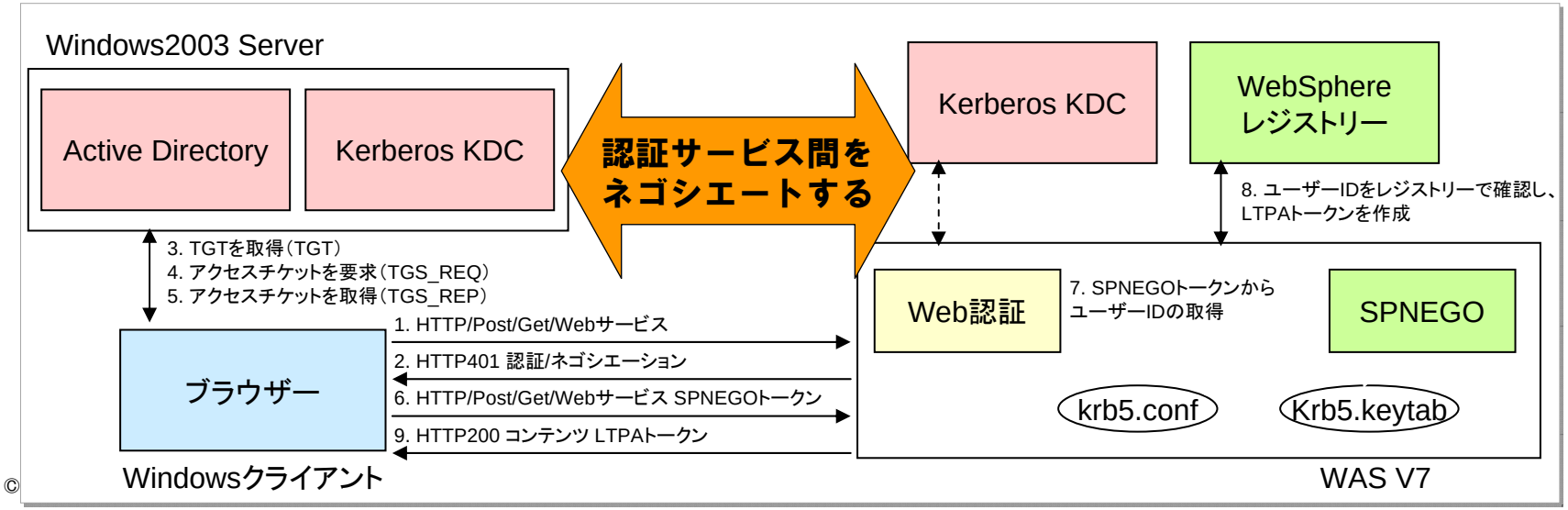
Kerberos認証

- クライアントとサーバー間の通信を暗号化するユーザー認証メカニズム
  - ギリシア神話に登場する地獄の番犬に由来し、この怪物と同様にネットワークにアクセスする者の認証を行う
- WAS V6.1よりKerberos認証をサポート
  - Windowsを始めとした様々な製品とシングル・サインオンが可能に
- Kerberos認証の挙動
  - ①ネットワークへのユーザー・ログオン
    - KDC (Key Distribution Center)にアクセスして認証を受ける
      - Windows ネットワークでは、Kerberos認証機能が標準で実装されているため、ドメイン・コントローラがKDCとして機能する
    - 認証が成功するとクレデンシャルがKDCからユーザーに付与される
      - クレデンシャルはセッション鍵とTGT(Ticket-Granting Ticket:チケット発行のための大元のチケット)を含むデータである
  - ②リソースへのアクセス（セッション鍵によりすべて暗号化されている）
    - リソースを指定し、KDCにTGTを提示する
    - KDCはTGTに記載されている有効期限などをチェックする
    - KDCはリソースにアクセスするための「アクセスチケット」をユーザーに返す
    - ユーザーはアクセスチケットを利用し、リソースにアクセス可能となる

SPNEGO Web認証

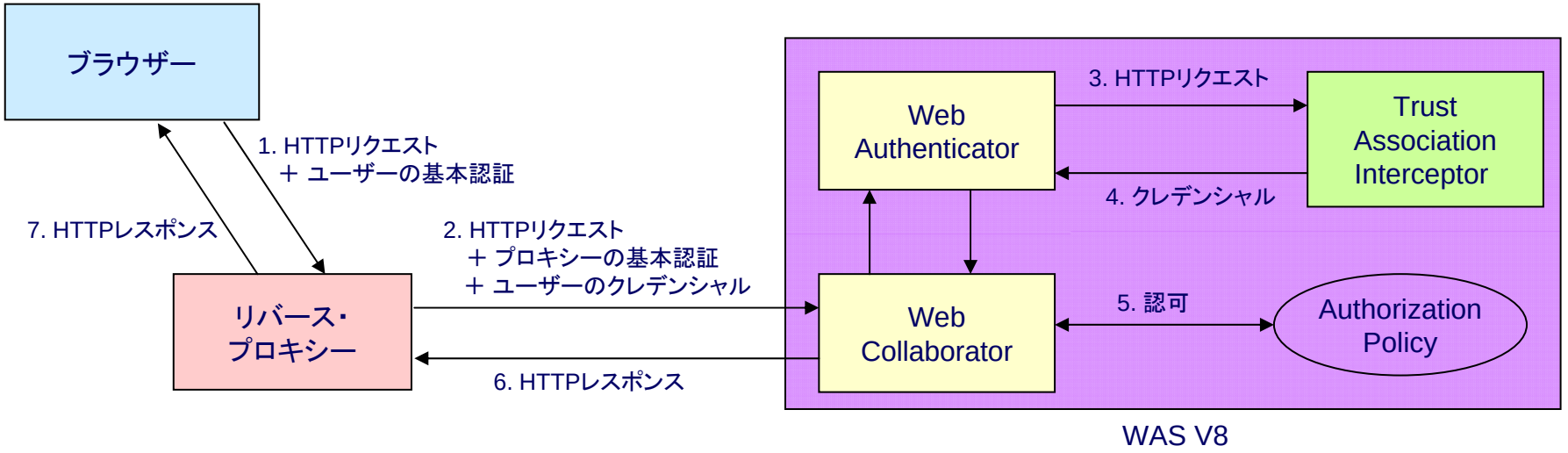
- IETF RFC 2478で定義された認証メカニズム
  - Simple and Protected GSS-API Negotiation Mechanismの略
  - WASはSPNEGOを使用することで、Windows-WAS間の統合シングル・サインオン環境を確立する
- SPNEGO TAIは非推奨になり、WAS V7.0からSPNEGO Web認証に変更
  - WAS V7.0以降での拡張
    - 管理コンソールからの設定が簡便化
    - SPNEGOランタイムの動的更新
    - セキュリティー・ドメイン・レベルでカスタマイズが可能
    - アプリケーション・ログイン・メソッドへのフォールバックが可能

・単一 Kerberos レルム内でのSPNEGO Web認証



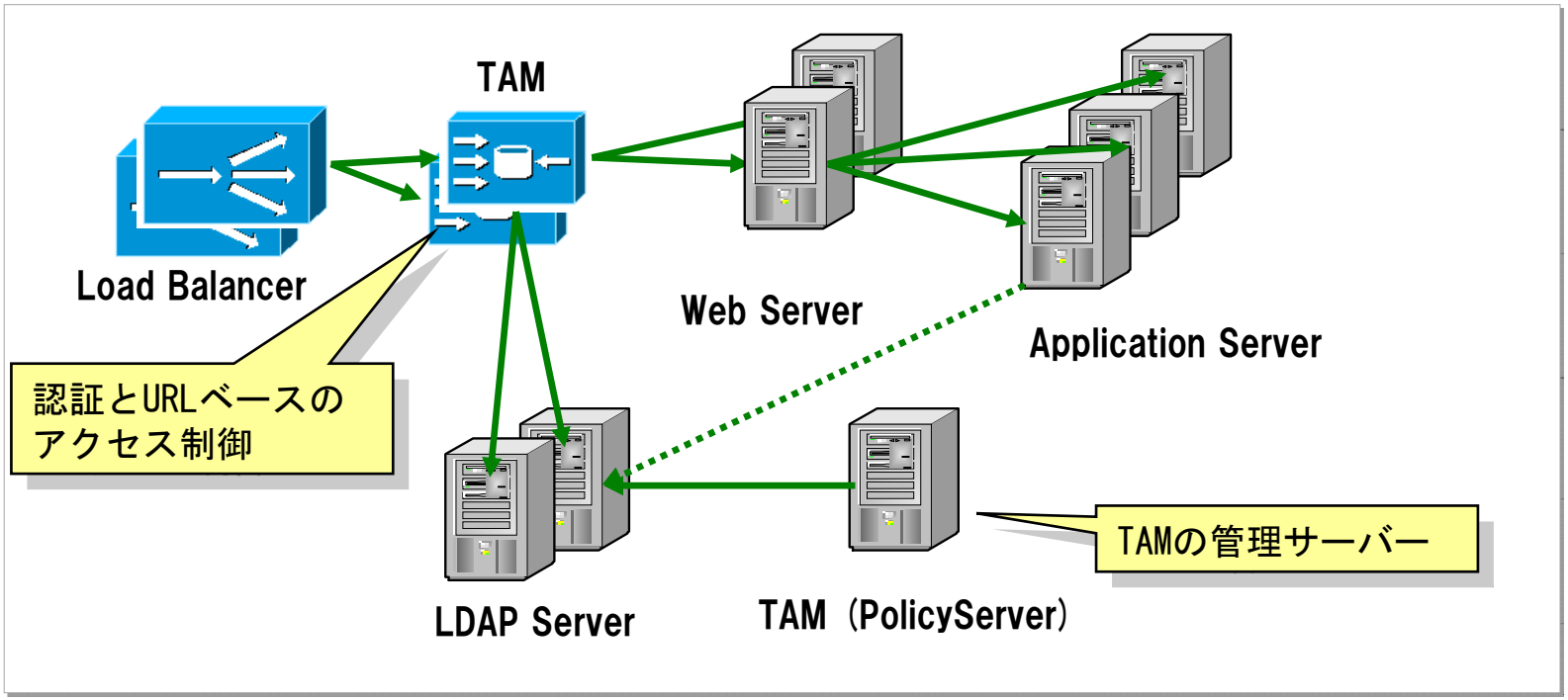
トラスト・アソシエーション

- 前段のリバース・プロキシーを信頼しユーザー情報を受け入れる
  - トラスト・アソシエーション・インターセプター（TAI）が、HTTPリクエストからユーザー情報を取得
  - 提供されている TAI
    - com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus
      - WebSEAL V5.1 用⇒TAM連携
    - com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl
      - SPNEGO TAI 用（非推奨）



【参考】TAMと連携したWebシステム構成

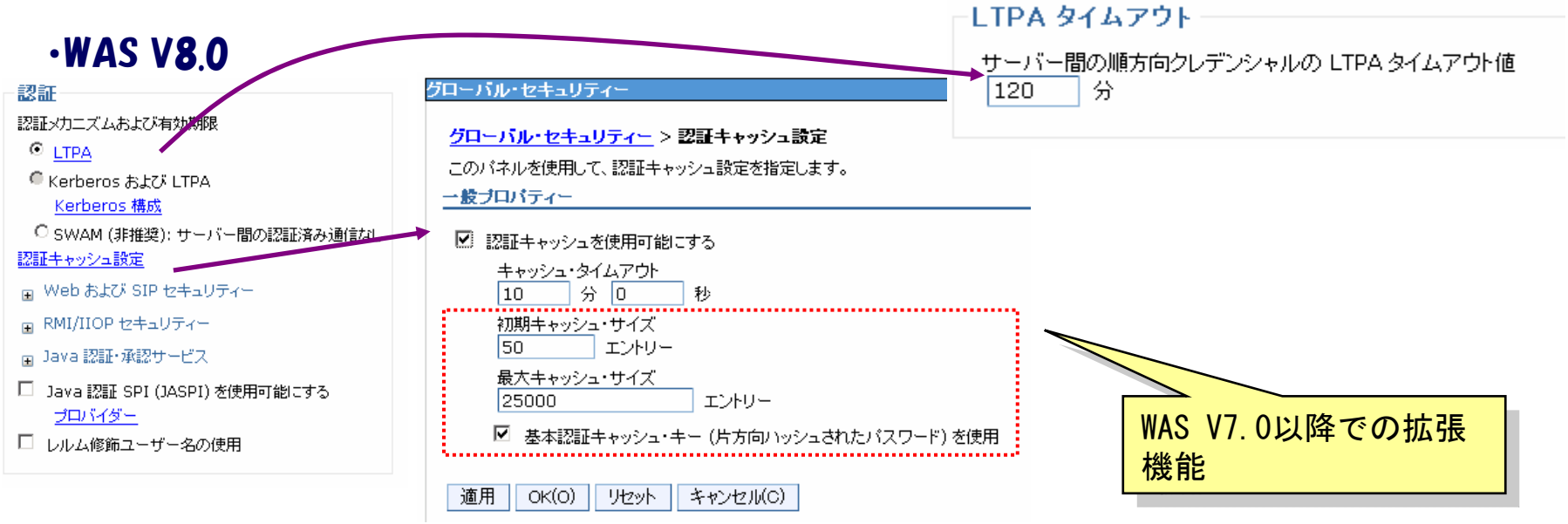
- 認証をTivoli Access Manager (TAM)で行う構成
  - バックエンドシステム(Webサーバー、ホスト等)の認証機能をTAMで代行し、集約することでセキュリティー・レベルをTAMで一括管理する
    - TAMでは、認証と大まかなアクセス制御を行う
    - バックエンドのシステムでは、詳細なアクセス制御を行う



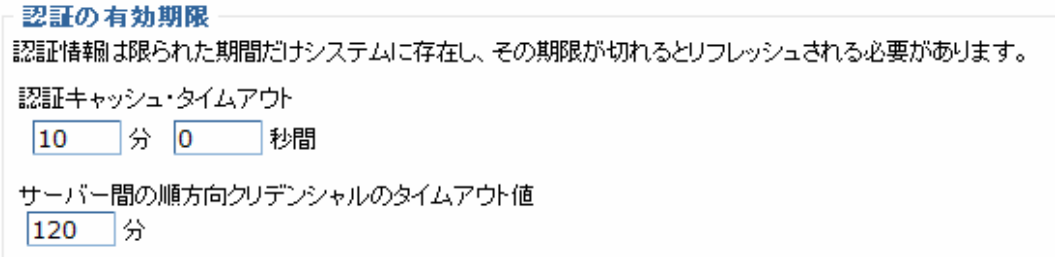
認証キャッシュ設定

- 認証キャッシュ設定
  - WAS V7.0以降、一部のパラメーターが管理コンソールから設定可能
  - パフォーマンスの観点から、使用可能に設定することが推奨

・WAS V8.0



・WAS V6.1



2. WASセキュリティー設計

2-1 ハードニング
2-2 認証
2-3 認可
2-4 暗号化 / 署名
2-5 監査

認可とは、保護対象リソースに対して適切なアクセス権限を付与すること

アプリケーション・セキュリティー：ロールベースの認可

- 認可： 前提として認証が必要（アクセス管理 = 認証 ＋ 認可）
- 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もっているかを判定する
  - 管理セキュリティーとアプリケーション・セキュリティーが対象

- Java EE の標準の認可方式
  - ロール： アプリケーション開発者が定義
  - リソース許可： アプリケーション開発者が定義
  - ロール・マッピング： デプロイ時にロールをユーザー/グループへマップ

例： Teller ロールを持つ Teller Grp や Frank は、Account Bean のメソッド getBalance と deposit を実行可能

【リソース許可】
リソースとロールのマッピング表

		Account Beanのメソッド		
		getBalance	deposit	closeAccount
ロール	Teller	○	○	
	Cashier		○	
	Supervisor			○

【ロール・マッピング】
ロールとユーザー/グループのマッピング表

		ロール		
		Teller	Cashier	Supervisor
グループまたはユーザー	Frank	○		○
	Teller Grp	○		
	Cashier Grp		○	
	Supervisor Grp			

アプリケーション・セキュリティー： JACC を使用した認可

- WASのセキュリティー・ランタイムの代わりに、
  JACC（Java Authorization Contract for Containers）プロバイダーが認可
  - JACC プロバイダーの例： Tivoli Access Manager
- セキュリティー・ドメイン・レベルでカスタマイズが可能

管理ロール

- WAS管理操作に対して、アクセス制御を提供する機能
  - 権限は管理ロールとして定義され、1ユーザー、1グループに対し複数の管理ロールを設定可能

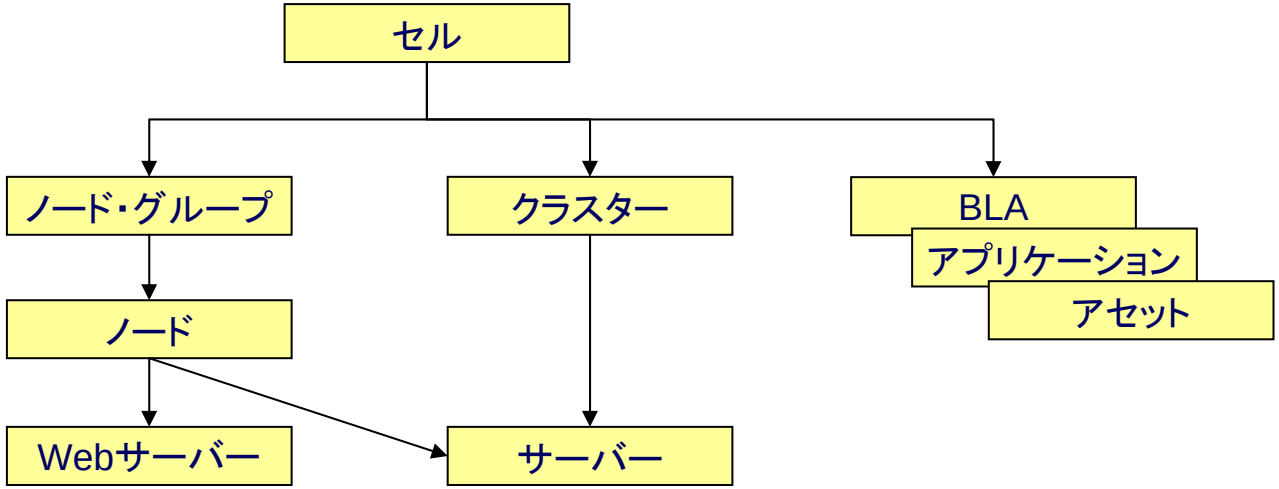
※ WAS一次管理者に割り当てられる管理ロール

管理ロール名	説明
管理者※	オペレーター、コンフィギュレーターに与えられる権限および管理者ロールのみに許可されている追加権限を持つ。
セキュリティー・マネージャーの管理※	管理者ロールを割り当てることができる。また、詳細な(Fine-grained)管理セキュリティー使用する際に、このロールを持つユーザーだけが許可グループを管理できる。
監査員※	セキュリティー監査の確認、設定変更権限を持つ。
コンフィギュレーター	モニターに与えられる権限を持ち、WASの構成を変更することができる。
オペレーター	モニターに与えられる権限を持ち、起動・停止といったランタイムの状況を変更することができる。
デプロイヤー	アプリケーションに関する変更およびランタイムの操作をおこなうことができる。
ISC管理者	管理コンソールのみ使用可能で、フェデレーテッド(統合)リポジトリー構成のユーザーおよびグループを管理することができる。
モニター	WASの構成および現在のランタイムの状況を確認することができる。

管理ロールの設定方法は、「セキュリティー設計-参考-」のP.89をご参照下さい。

管理セキュリティー： 詳細な（Fine-grained ）管理セキュリティー

- よりきめ細やかな管理セキュリティー機能を提供
- ユーザーの管理ロールを特定のリソース・インスタンスに対してのみ割り当てることが可能
  - 管理ロールは、セル内の全リソースが対象
  - 詳細な（Fine-grained ）管理セキュリティーは、管理対象リソースを用途ごとにまとめた「管理許可グループ」が対象
  - 割り当てられたリソース以外にアクセスすることはできない
  - セキュリティー・マネージャーの管理ロールをもつユーザーが許可グループを管理する
    - リソース・タイプとして、クラスター、サーバー、Webサーバー、ノード・グループ、ノード、ビジネス・レベル・アプリケーション(BLA)、アプリケーション、アセットを対象とした管理ロールを割り当てることが可能
    - リソース・タイプには親子関係があり、上位のリソースに対して権限を付与するとその下のリソースに対する権限が与えられる



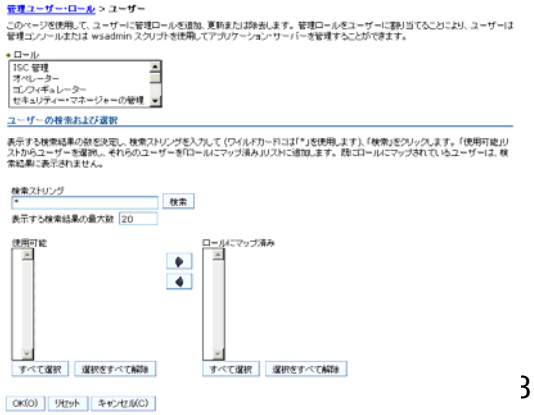
管理ロールの設計指針

- 前提
  - ユーザーに対して、複数の管理ロールを割り当てることが可能
  - グループに対して、複数の管理ロールを割り当てることが可能
    - グループ内のユーザーすべてにロールが付与される
- 設計指針
  - 個々のユーザーではなく、グループに対して管理ロールを付与する

① 許可検査時のパフォーマンスが向上する
通常は、ユーザー数よりグループ数の方がはるかに少ない

② アクセス制御にグループのメンバーシップを使用することにより、柔軟性が向上する

③ 製品環境の外側で、グループにユーザーを追加したり、グループからユーザーを削除したりできる



【参考】アプリケーション・セキュリティーの認証・認可を行うコンポーネント

	Webサーバー	WAS	TAM
認証方法	ベーシック認証 ダイジェスト認証 クライアント証明書	ベーシック認証 フォーム認証 クライアント証明書	ベーシック認証 フォーム認証 クライアント証明書 など
認可レベル	URL	宣言型：URL プログラム型：自在	URL
ユーザー・ディレクトリー	テキストファイル LDAP	ファイル LDAP データベース カスタム	LDAP
WASグローバル・セキュリティーの設定	不要	必要	不要
シングル・サインオン	–	可能 （LTPAトークンで認証情報を渡す）	可能 （LTPAトークン、BAヘッダーなどで認証情報を渡すことが可能）

認証はTAMで行い、細かなアクセス制御はWASで行う構成も可能

2. WASセキュリティー設計

2-1 ハードニング
2-2 認証
2-3 認可
2-4 暗号化 / 署名
2-5 監査

暗号化とは、通信経路等の暗号化を行うことにより保護対象リソースの盗聴を防ぐこと
署名とは、デジタル署名等により保護対象リソース(データ)の改ざんを防ぐこと

SSL (Secure Sockets Layer)

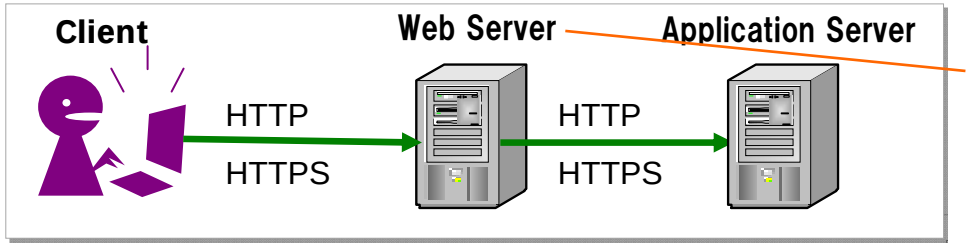
- サーバーとクライアント間のセキュアー接続を提供する技術
- トランスポート層において機能するプロトコル
  - 認証性　　クライアント認証、サーバー認証によるなりすまし防止
  - 保全性　　データ署名による改ざん防止
  - 機密性　　暗号化による盗聴防止

- X.509証明書
  - SSL実装には、Java Secure Sockets Extension (JSSE)を使用
  - X.509 証明書ベースの非対称の鍵ペアを信頼する
  - X.509証明書の内容
    - 署名、バージョン、シリアル番号、署名アルゴリズム識別子、発行者名、有効期間、サブジェクト名、証明書で公開鍵が識別されているエンティティーの名前、サブジェクトの公開鍵情報
  - X.509証明書の内容の署名
    - 認証局(CA)、ルート証明書、自己署名

SSLハンドシェイクは、「セキュリティー設計-参考-」のP.93をご参照下さい。
SSLハンドシェイクのキャッシュは、「セキュリティー設計-参考-」のP.94をご参照下さい。

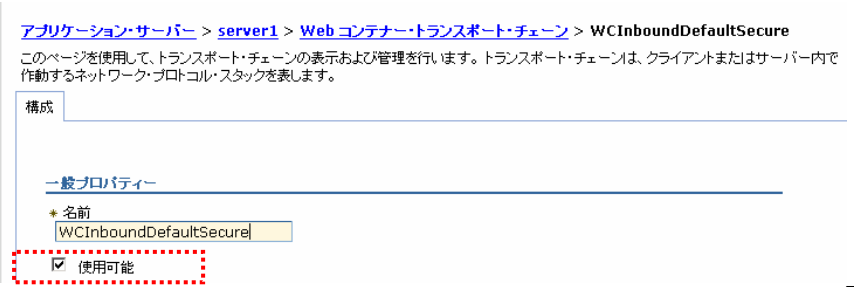
WASを使用したシステムでのSSL通信

- SSL通信の設定
  - IHS-WAS間は、クライアント-IHS間がSSL通信の場合、デフォルトでSSLで通信となる



・httpd.conf抜粋
<VirtualHost *:443>
SSLEnable
SSLServerCert selfSigned
</VirtualHost>
KeyFile
"/opt/IBM/HTTPServer70/conf/ihskeys.kdb"

- IHS-WAS間のSSL
  - IHSとWASが同一筐体の場合は、パフォーマンスの観点から解除することを検討する
  - 設定方法
    - 管理コンソールより、アプリケーション・サーバー名 >- Webコンテナー設定 >- Webコンテナー・トランスポート・チェーンを選択し、WCInboundDefaultSecureのトランスポート・チェーンをOFFにする



WASの各コンポーネント間で使用されるSSL構成は、「セキュリティー設計-参考-」のP.95をご参照下さい。

WASでのSSL実装

- WASは、JSSEを用いてSSLを実装する
  - JSSE（Java Secure Socket Extension）：SSLを実装するためのJava標準API
- JSSEでは、サーバー証明書と鍵を**キーストア・トラストストア・ファイル**で管理する
  - 証明書は、X.509 証明書 を使用
  - WASは、PKCS12 フォーマットを使用（JKS, JCEKS, PKCS12 の3種類のフォーマットをサポート）
  - IHS、プラグインはKDBフォーマットを使用しており、PKCS12とは互換性がないため管理の際は注意
- WAS管理セキュリティーでは、Javaプロセス間の双方向SSL通信を実施する
  - サーバー認証とクライアント認証の両方を実施

キーストア

–自分の証明書、鍵などの機密情報のリスト

トラストストア

–機密情報を含まない

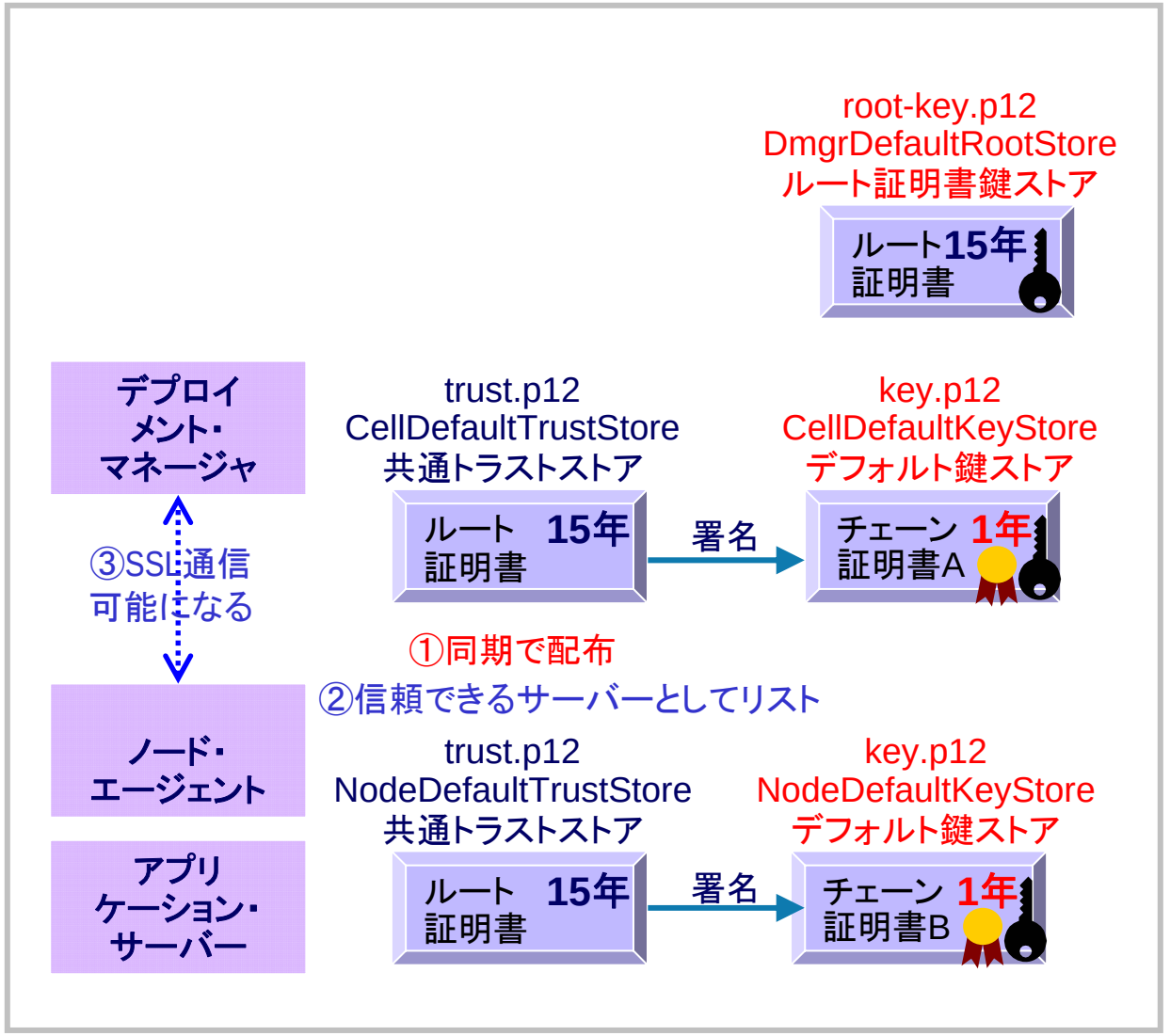
–SSL通信したい相手の証明書のリスト

それぞれのファイル名はP54を参照

SSL通信のためには、
相手のサーバーを信頼する＝トラストストアに相手の証明書がリストされている必要がある

【参考】WASが使用するキーストア・トラストストア

重要

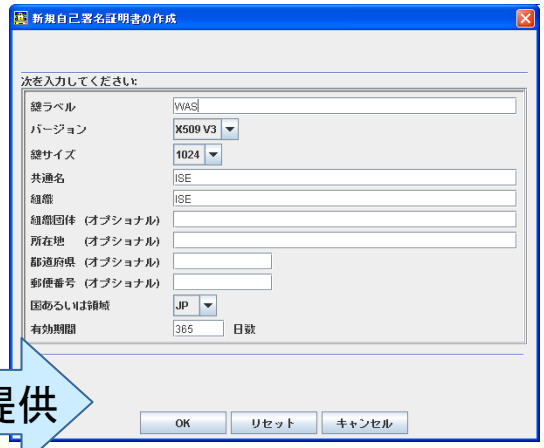


証明書と鍵の管理

- 証明書有効期限の管理
  - WASによって管理されるキーストアに格納された証明書の期限を監視、期限切れ前に警告を出す
    - 自己署名証明書とチェーン証明書は、自動更新することも可能
- IBM 鍵管理ユーティリティー (iKeyman)と同等の機能が、管理コンソールから使用可能に
  - ikeymanは引き続き利用可能
  - 証明書要求は生成したツールを使用して受信する必要がある

・管理コンソール

・ikeyman



同じ機能が提供

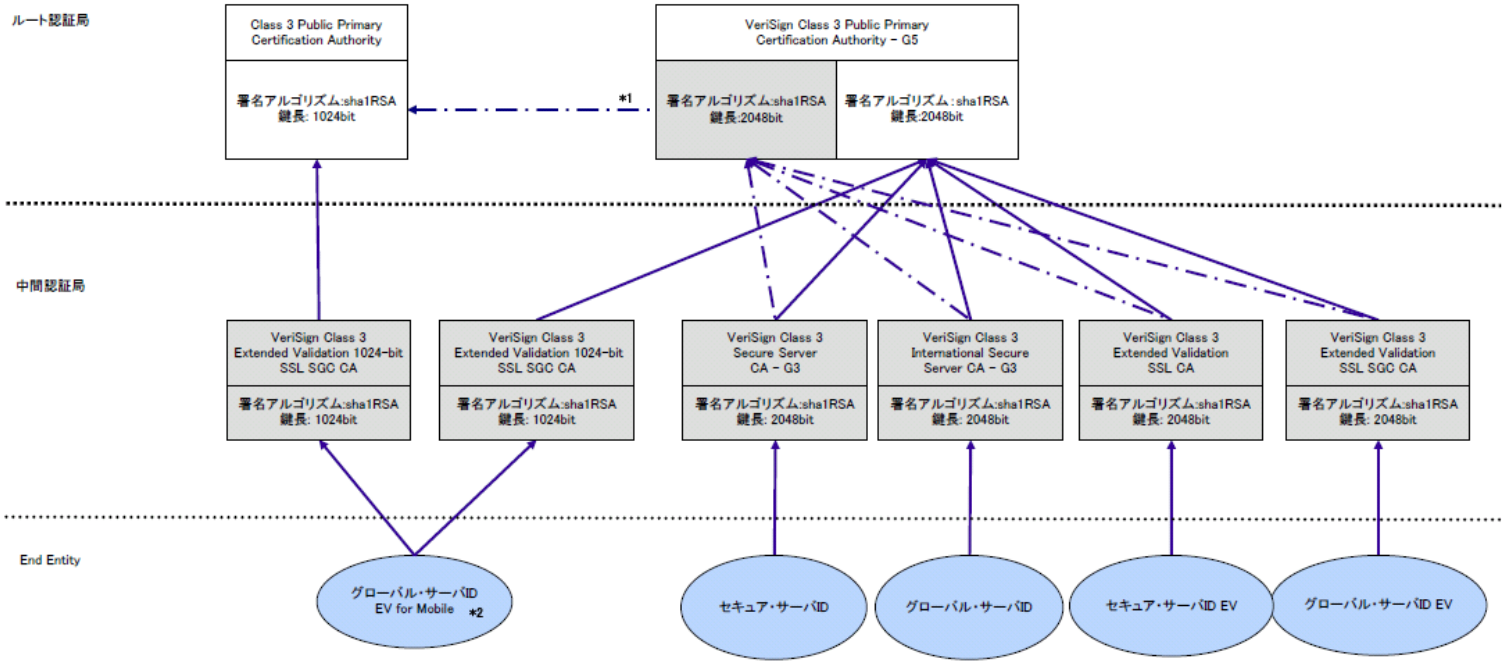
・新しいキー・データベースを作成する
・新しい鍵のペアと認証要求の作成
・自己署名付き証明書の作成
・認証要求の再作成
・CA署名付き証明書をキー・データベースに受け取る
・CAのルート証明書を保管する
・キー・データベースをオープンする
・鍵を別のデータベースまたはPKCS12ファイルにエクスポート・インポートする
・キーの抽出・削除
・認証局(CA)と認証要求をリストする
・キー・データベースのデフォルト鍵を作成する
・暗号化されたデータベース・パスワードをstashファイルに保管する

【参考】チェーン証明書の仕組み

確認

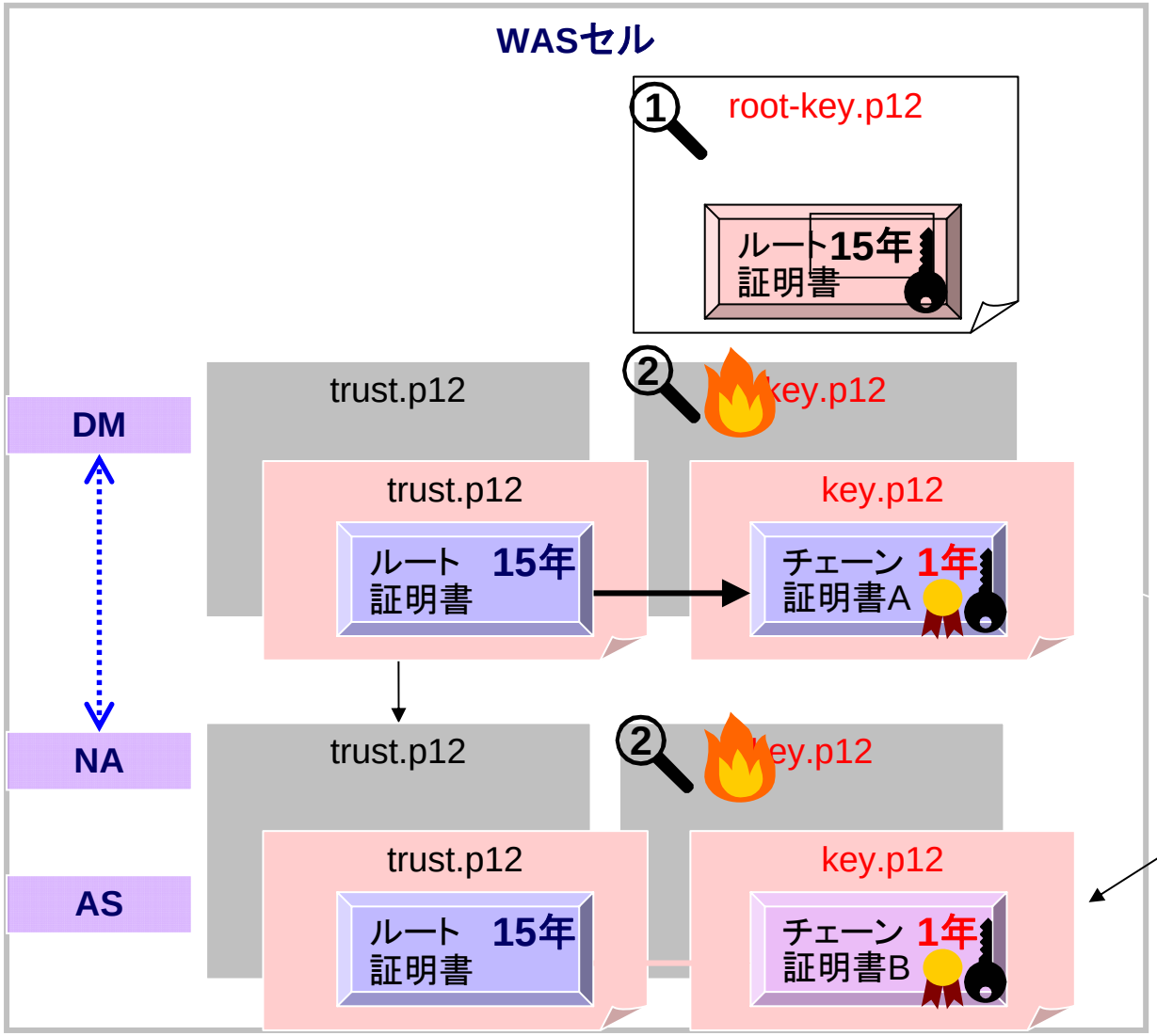
チェーン証明書の仕組み

- 通信を行う両者の間でルート証明書を安全に共有する
  - ■ブラウザーなどでは、大手認証局のルート証明書があらかじめインストールされている
- 実際に使用する証明書は，ルート証明書で電子署名されている
  - ■あるいは，「ルート証明書」で電子署名された「中間証明書」で、電子署名 される
- 署名を順にたどって，最終的にルート証明書に行き着けば，その証明書は信頼できると判断する



【参考】証明書の有効期限切れと自動更新

重要



STEP 1
ルート鍵ストアの検証

STEP 2
残りの鍵ストアの検証

STEP 3
証明書の有効期限切れが検知される

STEP 4
署名者証明書ごと置き換えられる

STEP 5
新規証明書が同期して各ノードに配布される

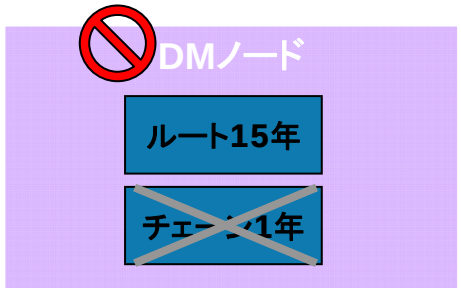
! 同期が重要

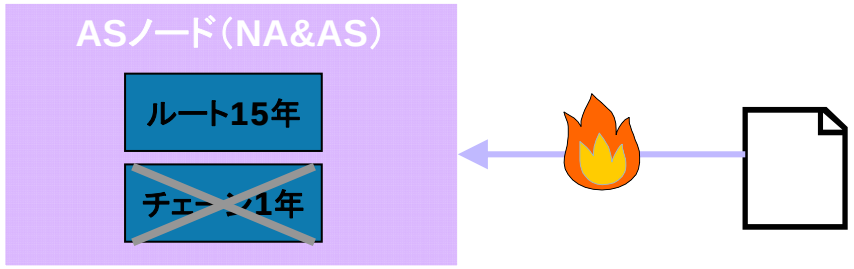
【参考】証明書の有効期限切れ問題

事例

同期に失敗して問題となったケースと対応

- 運用方法
  - 管理セキュリティーはOn
  - 証明書の有効期限はデフォルト
  - 証明書自動更新はOn
  - DMは常時停止
  - 自動同期はOff
  - 運用シェルでプロセスを起動停止





- 現象
  1. 証明書の有効期限が切れる
  2. アプリケーション・サーバーを停止しようとすると、証明書関連のプロンプトが出力される
  3. 停止運用シェルの処理が上記②で一時停止し、アプリケーション・サーバーが停止したようにみえる
- 対応
  1. DMを起動
  2. wsadminコマンドで証明書更新
  3. 同期実施
  4. アプリケーション・サーバーを手動で一旦停止・起動

WASでのSSL証明書　設計のポイント

- プロファイル作成時に、証明書の有効期限を長く設定する　重要
  - ルート証明書　15年（デフォルト）～25年　を指定可能
  - チェーン証明書　1年（デフォルト）～15年　を指定可能

- プロファイル作成後に、証明書置き換えも可能
  - 手順は下記ガイド参照
    - 【FAQ】WAS V7.0 個人証明書（チェーン証明書）の置き換え手順
    - http://www.ibm.com/jp/domino01/mkt/websphere.nsf/doc/006170D2

- 証明書の有効期限をモニター機能有効化を検討する
  - 以下の条件であれば自動更新に問題なし
    - DMは常時起動している
    - 同期は自動同期
  - DMを常時停止している場合、自動同期を無効化している場合は、更新手順を確認
    - 【考慮事項】WAS ND V7.0 証明書自動更新機能について (WAS-09-024)
    - http://www.ibm.com/jp/domino01/mkt/cnpages1.nsf/page/default-0007F8D1

暗号アルゴリズムの2010年問題とは

- 米国立標準技術研究所（NIST）が現在の標準暗号の利用を廃止し、より安全性の高いであろう次世代暗号への2010年内に移行を強制する方針を打ち出したこと が発端

暗号技術要素	移行対象となった規格	推奨される規格
共通鍵暗号	2-key Triple DES（2TDES）	AES
公開鍵暗号	1024-bit RSA/DH/DSA	2048-bit RSA/DH/DSA以上 or 256-bitECDSA/ECMQV以上
ハッシュ関数	SHA-1	SHA-2（SHA-224以上）

但し日本では2010年から「要件」とし、2013年までに移行を「完了」するスケジュール

- CA/ブラウザ・フォーラムではNISTの方針を受け，EV-SSL証明書について，2010年12月末までに1024bit鍵長の加入者(End-Entity)証明書を失効させるよう決定

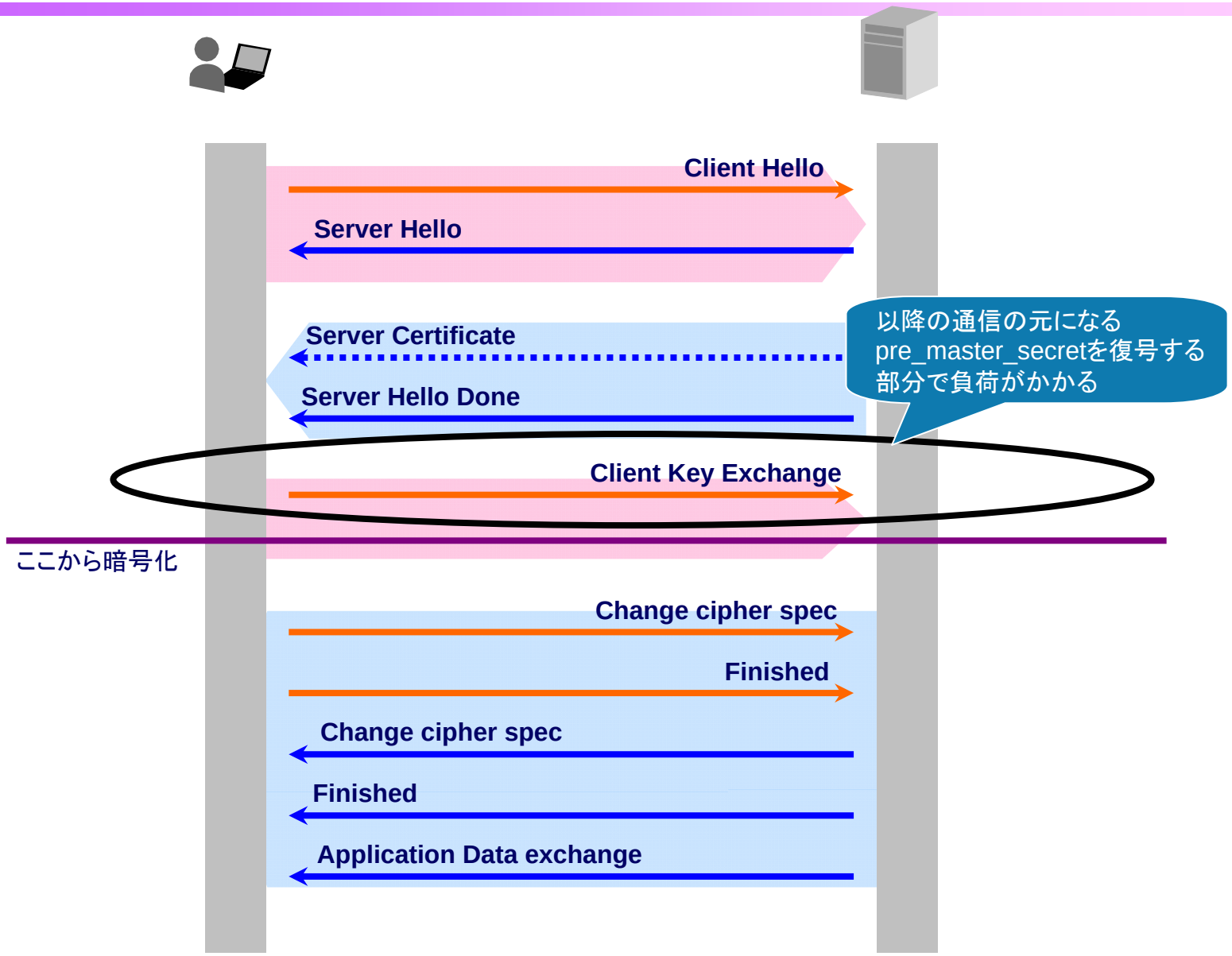
  → 2010年以降は2048bit鍵長のEV-SSL証明書を使用を強制

- さらに、ベリサインでは、グローバル・サーバーIDおよびセキュア・サーバーIDについても2010年内に1024bitから2048bitに仕様変更する旨を公表

2010年問題への対応

- ベンダーへ置き換えの要否を確認する
  - 証明書には信頼性に応じて複数の証明書がある
  - EV-SSL証明書
    - 2010年中に、暗号化強度1024ビットから2048ビットへの置き換えが必須
  - その他の証明書
    - 各ベンダーによって対応が異なるため、ベンダーへ置き換えの要否を確認する
    - 現状ではベリサインが早くも他の規格の証明書でも2048bit化を推進を表明
- IHSでサーバー証明書の置き換えが必要となった場合
  - サーバーの暗号化・復号処理のためのCPU負荷が増大するため、検証環境等で負荷を確認する
  - チューニング・パラメーターを見直す
- 証明書を交換した場合、古い証明書は削除

【参考】SSLハンドシェイクの2048bit化で影響を受ける箇所



【参考】2048bitの負荷の極小化のポイント(1/2)

- KeepAlive（デフォルト On）
  - 接続を使いまわすことができれば，余計なSSLハンドシェイクを行わずに済む
- KeepAliveTimeout （デフォルト15秒）
  - KeepAliveTimeoutで設定した期間中はスレッドが占有されてしまうので、1～2秒程度の短い値を設定
- MaxKeepAliveRequests(デフォルト100)
  - 平均コンテンツ数程度が推奨
- SSLV3Timeout （デフォルト120秒）
  - SSLセッションIDを有効にして，不要にClientKeyExchangeを有するSSLハンドシェイクを行わないようにする
  - セキュリティー低下(なりすまし)につながる恐れもあるため，長時間の値は設定しない
  - 120秒を超える値を設定した場合，クライアントが使用するInternet Explorerのバージョンによっては，120秒以内にブラウザーがSSLセッションIDを破棄するため，SSLV3Timeoutで設定した値が有効にならない場合がある
- GSK_V3_SIDCACHE_SIZE（OSの環境変数）
  - Windows系あるいはSSLCacheDisableを明示的に記述した場合、GSKitでSSLセッションIDをキャッシュする
  - デフォルト512エントリーで，それを超えるリクエストがある場合，超えた分のSSLセッションIDはキャッシュされないので、注意が必要
  - 1～4096まで設定可能なので、システムの同時接続数から設定値を見積もる

【参考】2048bitの負荷の極小化のポイント(2/2)

分散環境の場合は以下も考慮する必要がある

- Affinity
  - SSLセッションIDはサーバー固有のものであるため，負荷分散装置により接続ごとに異なるサーバーに割り振られる場合，その都度新規のSSLハンドシェイクを実施してしまうので、負荷が高まってしまう
  - Edge Components等でアフィニティーを設定できる場合はなるべく設定する

- ETag（デフォルトではiNodeを含む情報でキャッシュ判断）
  - デフォルトではサーバー依存のパラメーターをEtagに含んでしまうため、別サーバーに割り振られた場合に不要なキャッシュ・ミスが生じてしまう
  - バージョン6以下のIEはキャッシュ・ミス時にRSTパケットを返すため、後続のリクエストは再度SSLハンドシェイクを行うことになる(IE7以上, Firefox, Google Chromeは発生しない)

2. WASセキュリティー設計

2-1 ハードニング
2-2 認証
2-3 認可
2-4 暗号化 / 署名
2-5 監査

監査とは、保護対象リソースに対して、いつ、誰が、何をしたのかを確認すること

セキュリティー監査

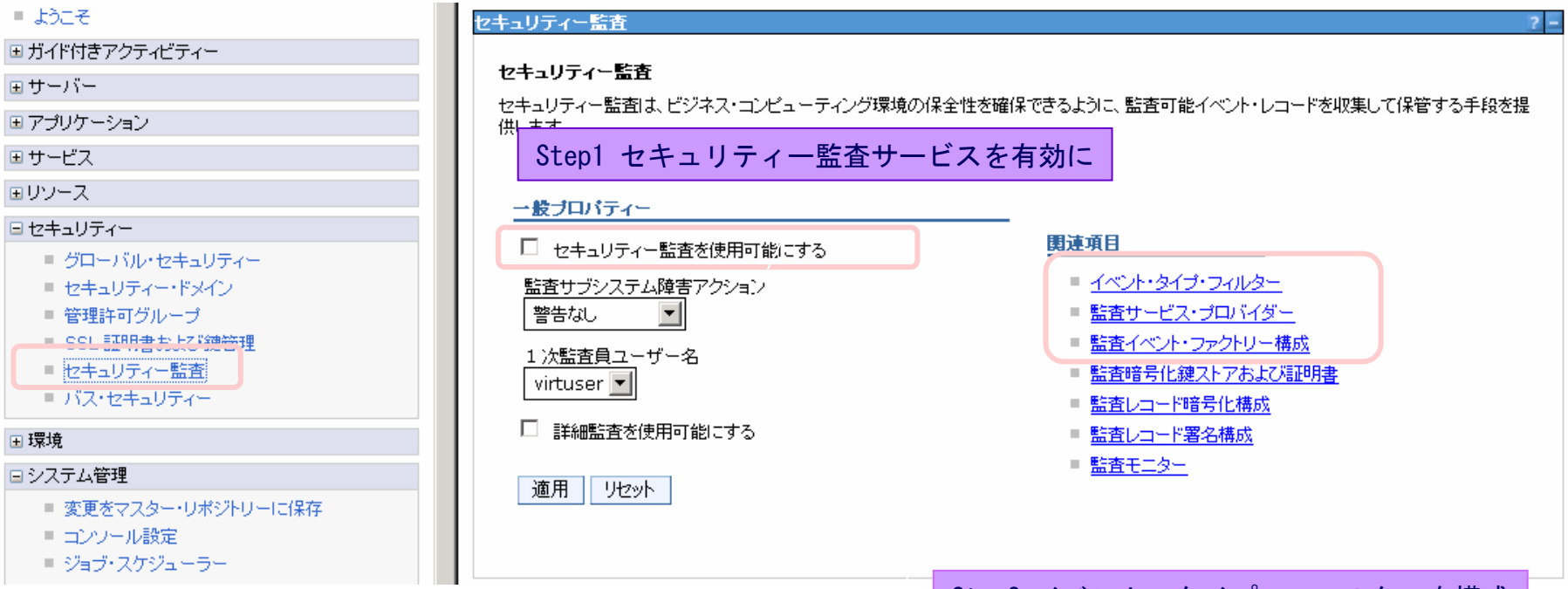
- セキュアーIT環境の統制に利用できる監査記録を提供
  - WASシステムインフラ、アプリケーションで認証・認可やその他セキュリティー・イベントをモニター
  - 「いつ」、「誰が」、「何をおこなったのか」を監査ログに記録
  - 法令遵守を証明するための仕組みや脆弱性分析に利用可能

- セキュリティー監査機能の特徴
  - グローバル・セキュリティーを有効にすることが前提
  - 監査ログファイルはバイナリー保存
    - 監査ログ・データをHTML形式のレポートに変換するAudit Readerが提供されている
    - ログはデータの暗号化、署名による保護が可能
    - *Update v8* V8から監査ログ・ファイルの最大数に達したときの動作を指定可能に
      - 最も古いものを上書き / サーバーの停止 / ロギングの停止
  - 管理者(Administrator)と監査員(Auditor)の権限を分離
    - 管理者権限では監査記録やポリシーの表示および変更ができない
    - 監査員権限ではWASの構成やランライムの変更ができない
  - サード・パーティー製のセキュリティー監査サービスとの統合も可能

セキュリティー監査の設定

- セキュリティー監査は下記の画面で設定(設定後はJVMの再起動が必要)

  管理コンソール: セキュリティー > セキュリティー監査



Step2 イベント・タイプ・フィルターを構成
Step3 監査サービス・プロバイダーを構成
Step4 監査イベント・ファクトリーを構成

監査イベント・タイプ一覧(1/2)

イベント名	監査取得対象
SECURITY_AUTHN	全ての認証
SECURITY_AUTHN_CREDS_MODIFY	ユーザーIDに対するクレデンシャルの変更
SECURITY_AUTHN_DELEGATION	IDアサーションやRunAsなどの委任情報関連
SECURITY_AUTHN_MAPPING	2つのユーザーIDが含まれるクレデンシャル・マッピング情報関連
SECURITY_AUTHN_TERMINATE	タイムアウト、ログアウト等による認証の終了
SECURITY_AUTHZ	システムのアクセス・コントロール・ポリシー実行時の認証
SECURITY_ENCRYPTION	Webサービスの暗号化など暗号化情報
SECURITY_MGMT_AUDIT	監査の開始、停止などセキュリティー監査サブシステム操作関連
SECURITY_MGMT_CONFIG	WASの管理・構成操作関連
SECURITY_MGMT_KEY	証明書の作成、更新など証明書に対する管理操作関連
SECURITY_MGMT_POLICY	アクセス制御リストの作成等のセキュリティー・ポリシー関連
SECURITY_MGMT_PROVISIONING	ユーザー・アカウントの作成やグループへの追加等のプロビジョニング
SECURITY_MGMT_REGISTRY	ユーザー、グループの作成、変更など認証リポジトリー管理関連
SECURITY_MGMT_RESOURCE	ファイル、Webページなどのリソース属性の作成、削除等のリソース管理
SECURITY_RESOURCE_ACCESS	Webページやデータベースなどリソースに対する全てのアクセス
SECURITY_RUNTIME	起動、停止などWASランタイムのイベント
SECURITY_RUNTIME_KEY	有効期限切れなど証明書に対するランタイム操作関連
SECURITY_SIGNING	Webサービスに対するSOAPメッセージを有効にするためなどの署名操作関連

監査イベント・タイプ一覧(2/2)

- イベントに対する結果を指定する

結果名	説明
SUCCESS	対象イベントが成功したときに記録
FAILURE	対象イベントが失敗したときに記録
REDIRECT	対象イベント発生時にリダイレクトが発生したときに記録(例: SECURITY_AUTHNイベントが発生し、ログイン・ページやエラー・ページにリダイレクトする)
DENIED	対象イベントが拒否されたときに記録(例: リソース・アクセス時に適切な権限がない)
ERROR	対象イベントでエラーが発生したときに記録
WARNING	対象イベントでワーニングが発生したときに記録

- 設計指針
  - パフォーマンスへの影響を考慮し、イベント名と結果名にて取得項目を絞る

- 監査ログ結果のHTML化
  - バイナリー・ファイルであるため、Audit ReaderでHTML形式に変換して内容を確認する

```
AdminTask.binaryAuditLogReader('[-fileName <監査ログファイル名> -
outputLocation  <出力先ファイル名>  -reportMode complete ]')
```

セキュリティー監査設定方法は、「セキュリティー設計-参考-」のP.99～ P.102をご参照下さい。

監査ログの確認

- 監査ログはバイナリー･ファイルであるため、Audit Reader で HTML 形式に変換して内容を確認

◆wsadminコマンドプロンプトより、以下のコマンドを入力(下記はJython形式)

```
AdminTask.binaryAuditLogReader( '[-fileName 〈監査ログファイル名〉
                        -outputLocation 〈出力先ファイル名〉-reportMode complete ] ' )
```

◆実行例)

```
AdminTask.binaryAuditLogReader('[-fileName
/usr/IBM/WebSphere/AppServer/profiles/Dmgr01/logs/dmgr/BinaryAudit_W
ASCell01_WASCellManager01_dmgr.log -outputLocation /work/audit.html
-reportMode complete ]')
```

【参考】例：管理コンソールログイン・ログアウト

- セキュリティー要件
  - 管理コンソールにログイン・ログアウトしたユーザー情報を取得したい
    - ケース1：正常にログインできた場合
    - ケース2：不正にログインした場合（存在しないユーザー、パスワード間違い）
    - ケース3：正常にログアウトした場合

- 監査イベント/監査結果の設定

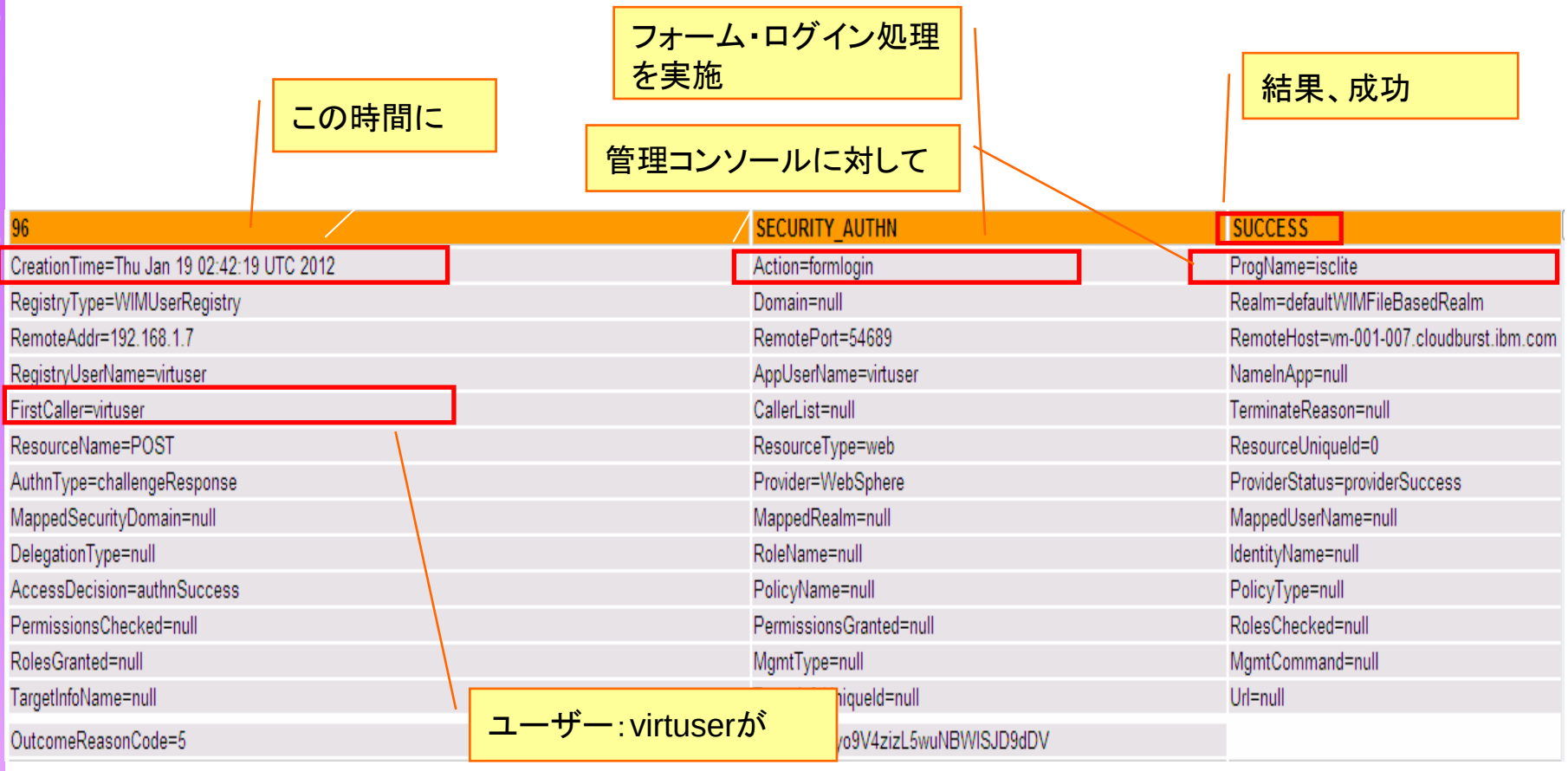
イベント名	監査取得対象
SECURITY_AUTHN	全ての認証を記録
SECURITY_AUTHN_TERMINATE	認証セッションの終了を記録（任意のログアウトやタイムアウト等によるログアウト）
SECURITY_RESOURCE_ACCESS	Webページやデータベースなどリソースに対する全てのアクセスを記録

結果名	説明
SUCCESS	対象イベントが成功したときに記録
REDIRECT	対象イベント発生時にリダイレクトが発生したときに記録（例：SECURITY_AUTHNイベントが発生し、ログイン・ページやエラー・ページにリダイレクトする）

/logonError.jspへリダイレクトする設定を事前に行う

【参考】ケース1：管理コンソールログイン・ログアウト結果

- 正常にログインできた場合の監査ログ出力
  - 監査イベント：SECURITY_AUTHN、監査結果：SUCCESSの出力を確認する



96	SECURITY_AUTHN	SUCCESS
CreationTime=Thu Jan 19 02:42:19 UTC 2012	Action=formlogin	ProgName=isclite
RegistryType=WIMUserRegistry	Domain=null	Realm=defaultWIMFileBasedRealm
RemoteAddr=192.168.1.7	RemotePort=54689	RemoteHost=vm-001-007.cloudburst.ibm.com
RegistryUserName=virtuser	AppUserName=virtuser	NameInApp=null
FirstCaller=virtuser	CallerList=null	TerminateReason=null
ResourceName=POST	ResourceType=web	ResourceUniqueId=0
AuthnType=challengeResponse	Provider=WebSphere	ProviderStatus=providerSuccess
MappedSecurityDomain=null	MappedRealm=null	MappedUserName=null
DelegationType=null	RoleName=null	IdentityName=null
AccessDecision=authnSuccess	PolicyName=null	PolicyType=null
PermissionsChecked=null	PermissionsGranted=null	RolesChecked=null
RolesGranted=null	MgmtType=null	MgmtCommand=null
TargetInfoName=null	niqueId=null	Url=null
OutcomeReasonCode=5	yo9V4zizL5wuNBWISJD9dDV	

【参考】ケース2：管理コンソールログイン・ログアウト結果

- 不正にログインした場合の監査ログ出力
  - 監査イベント：SECURITY_AUTHN、監査結果：FAILUREの出力を確認する

この時間に

フォーム・ログイン処理を実施

結果、リダイレクトされた

管理コンソールに対して

ユーザー：virtuserが

92	SECURITY_AUTHN	REDIRECT
CreationTime=Thu Jan 19 04:06:01 UTC 2012	Action=formlogin	ProgName=isclite
RegistryType=WIMUserRegistry	Domain=null	Realm=defaultWIMFileBasedRealm
RemoteAddr=192.168.1.7	RemotePort=54824	RemoteHost=vm-001-007.cloudburst.ibm.com
RegistryUserName=null	AppUserName=virtuser	NameInApp=null
FirstCaller=null	CallerList=null	TerminateReason=null
ResourceName=POST	ResourceType=web	ResourceUniqueId=0
AuthnType=challengeResponse	Provider=WebSphere	ProviderStatus=failure
MappedSecurityDomain=null	MappedRealm=null	MappedUserName=null
DelegationType=null	RoleName=null	IdentityName=null
AccessDecision=authnRedirect	PolicyName=null	PolicyType=null
PermissionsChecked=null	PermissionsGranted=null	RolesChecked=null
RolesGranted=null	MgmtType=null	MgmtCommand=null
TargetInfoName=null	TargetInfoUniqueId=null	Url=null
OutcomeReasonCode=15	SessionId=_r9OAL6iKjiuCjMPxgjmB3g	

  - 監査イベント：SECURITY_RESOURCE_ACCESS、監査結果：SUCCESSの出力にて、logonError.jspが表示されたことが確認できる

SECURITY_RESOURCE_ACCESS	SUCCESS
Action=resourceAccess	ProgName=/logonError.jsp
Domain=null	Realm=defaultWIMFileBasedRealm
RemotePort=null	RemoteHost=null
AppUserName=/UNAUTHENTICATED	NameInApp=null

【参考】ケース3：管理コンソールログイン・ログアウト結果

- 正常にログアウトした場合の監査ログ出力
  - 監査イベント：SECURITY_AUTHN_TERMINATE、監査結果：SUCCESSの出力を確認する

フォーム・ログアウト処理を実施

結果、成功

この時間に

管理コンソールに対して

105	SECURITY_AUTHN_TERMINATE	SUCCESS
CreationTime=Thu Jan 19 02:42:40 UTC 2012	Action=logout	ProgName=formlogin
RegistryType=WIMUserRegistry	Domain=null	Realm=defaultWIMFileBasedRealm
RemoteAddr=192.168.1.7	RemotePort=54689	RemoteHost=vm-001-007.cloudburst.ibm.com
RegistryUserName=null	AppUserName=virtuser	NameInApp=null
FirstCaller=virtuser	CallerList=null	TerminateReason=logout
ResourceName=GET	ResourceType=web	ResourceUniqueId=0
AuthnType=challengeResponse	Provider=WebSphere	ProviderStatus=providerSuccess
MappedSecurityDomain=null	MappedRealm=null	MappedUserName=null
DelegationType=null	RoleName=null	IdentityName=null
AccessDecision=logoutSuccess	PolicyName=null	PolicyType=null
PermissionsChecked=null	PermissionsGranted=null	RolesChecked=null
RolesGranted=null	MgmtType=null	MgmtCommand=null
TargetInfoName=null	TargetInfoUniqueId=null	Url=null
OutcomeReasonCode=9	=e8Pm5mWQWjh-R-TUgA5i2hQ	

ユーザー：virtuserが

まとめ・参考文献

まとめ

- WAS V8.0にて実現できる、Webシステムの脅威に対する対応策
  - ハードニング
    - ゾーニング、コンポーネントの制限、非rootユーザー稼動、IHSセキュリティー対策
  - 認証
    - グローバル・セキュリティー、複数セキュリティー・ドメイン、シングル・サインオン（SSO）
  - 認可
    - 管理ロール、詳細な（Fine-grained）管理セキュリティー
  - 暗号化 / 署名
    - SSL、証明書
  - 監査
    - セキュリティー監査
- セキュリティー強化　vs.　ユーザーの利便性向上
- セキュリティー強化　vs.　パフォーマンス向上

参考文献

- Information Center
  - WebSphere Application Server V8.0
    - http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp
- ワークショップ資料
  - WebSphere Application Server V8 アナウンスメント・ワークショップ
    - http://www.ibm.com/developerworks/jp/websphere/library/was/was8_ws/
  - WebSphere Application Server V7による基幹システム設計ワークショップ
    - http://www.ibm.com/developerworks/jp/websphere/library/was/was7_guide/index.html
- Redbook
  - IBM WebSphere Application Server V7.0 Security Guide
    - http://www.redbooks.ibm.com/abstracts/sg247660.html

Designing Web System Infrastructure with WAS V8.0

セキュリティー設計
– 参考 –

1. はじめに

・セキュリティー技術
・認証
・認可
・暗号化 / 署名
・監査

Java 2 セキュリティー

- ポリシー・ベースの精密なアクセス制御メカニズム
- 保護されるシステム・リソース
  - ファイル入出力
  - ソケット
  - ポート
- システム・リソースへのアクセスには、必ずアクセス権の検査が行われる
- ポリシー・ファイル
  - 静的ポリシー・ファイル
    - Javaプロセス用のセキュリティー・ポリシーを定義
    - WAS全体やサーバー全体に対する許可を設定するため、影響範囲が大きい
    - java.policy、server.policy、client.policy
  - 動的ポリシー・ファイル
    - エンタープライズ・アプリケーションのリソースおよびライブラリーのセキュリティー・ポリシーを定義
    - リソース単位に設定できるため、出来る限り動的ポリシー・ファイルにて制御を行う
    - spi.policy、 library.policy、 app.policy、was.policy、ra.xml

JACC (Java Authorization Contract for Containers)

- J2EE1.4で導入されたJava EEコンポーネントのひとつ(JSR115)
- Java EEコンテナーとJACCプロバイダーの間の実装規約
- WAS V7.0以降では、JACC仕様1.4が適用
  - セキュリティー・アノテーションの使用が可能
  - @DeclareRoles、@RunAs、@DenyAll、@PermitAll 、@RolesAllowed

- JACCプロバイダー
  - コンテナーとプロバイダー間の処理概要
    1. コンテナーがセキュリティー・ポリシー情報をプロバイダーに転送
    2. プロバイダーはそのセキュリティー・ポリシー情報を保管
    3. コンテナーはプロバイダーをアクセス許可を決定
- WASがサポートする2種類のプロバイダー
  - デフォルトのJACCプロバイダー
    - TAM(Tivoli Access Manager)のクライアント・サーバーの両方で実装され、NDパッケージの一部として付属
  - サード・パーティのJACCプロバイダー
    - JACC仕様で必要なポリシー・クラス、ポリシー構成ファクトリー・クラス、およびポリシー構成インターフェースを実装している必要がある

JAAS (Java Authentication and Authorization Service)

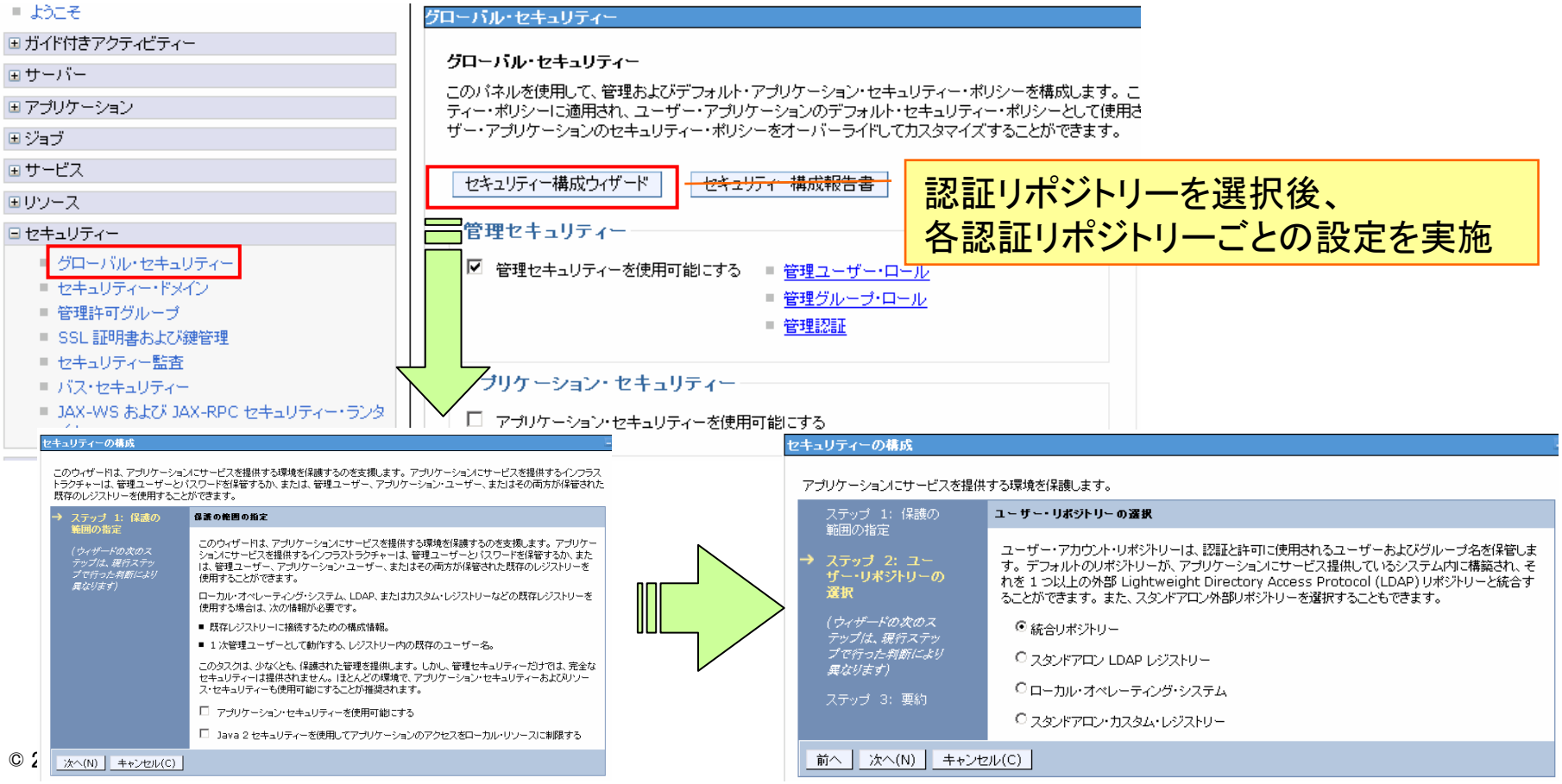
- Java 2 セキュリティー・アーキテクチャーの拡張
  - プラグ可能認証モジュール (PAM) 標準フレームワークの Java バージョンを実装
- PrincipalとSubjectによるアクセス制御を実施
  - Subject
    - １個以上のPrincipalから構成される
    - アクセス制御コンテクストに関連付けることができる
  - プリンシパル
    - ユーザーの識別情報

2. WASセキュリティー設計

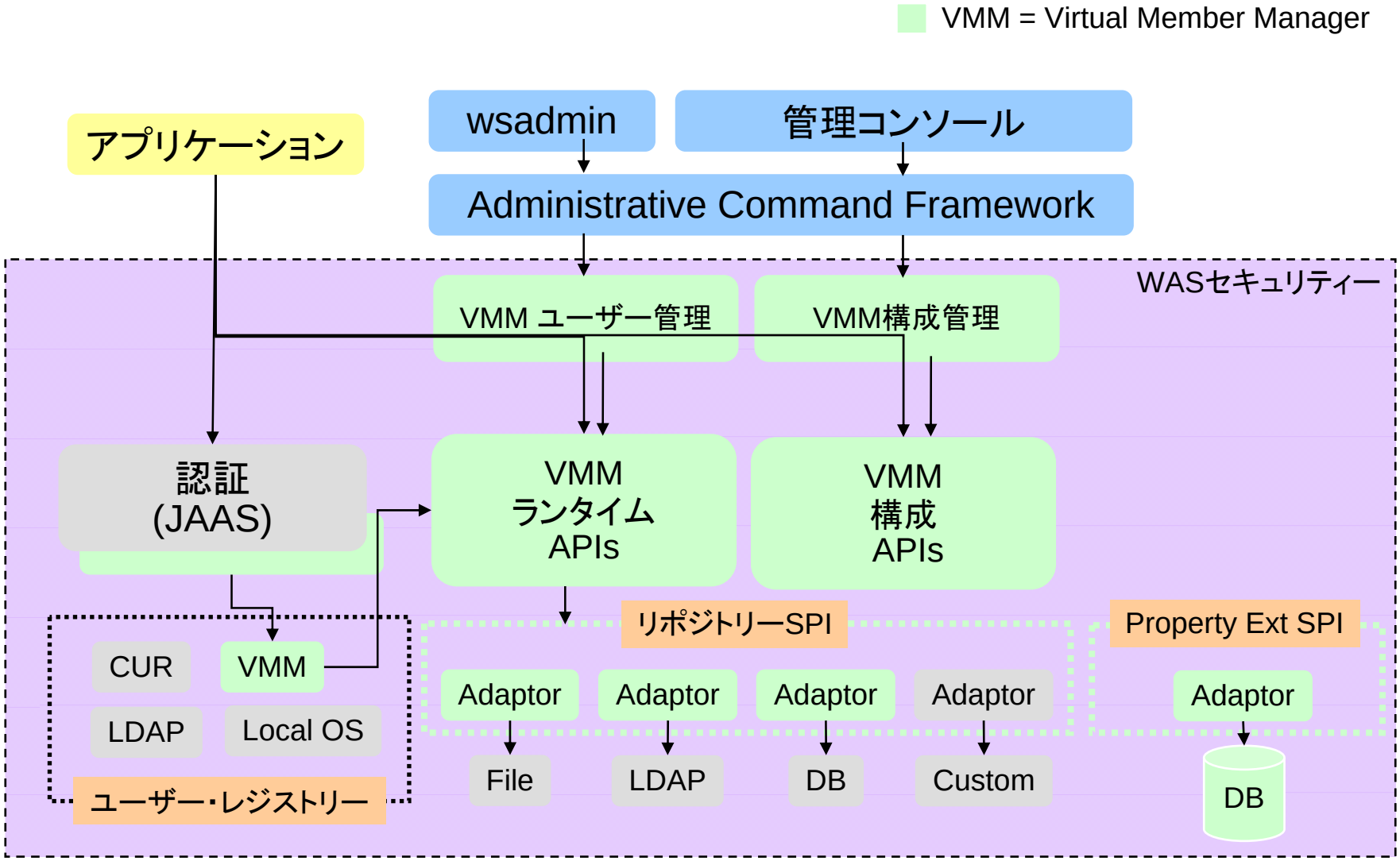
・セキュリティー技術
・認証
・認可
・暗号化 / 署名
・監査

グローバル・セキュリティーの設定方法

- セキュリティー構成ウィザード
  - プロファイル作成後はセキュリティー構成ウィザードを使用し、容易にグローバル・セキュリティーを設定可能
  - 管理コンソールより、セキュリティー > グローバル・セキュリティーを選択する

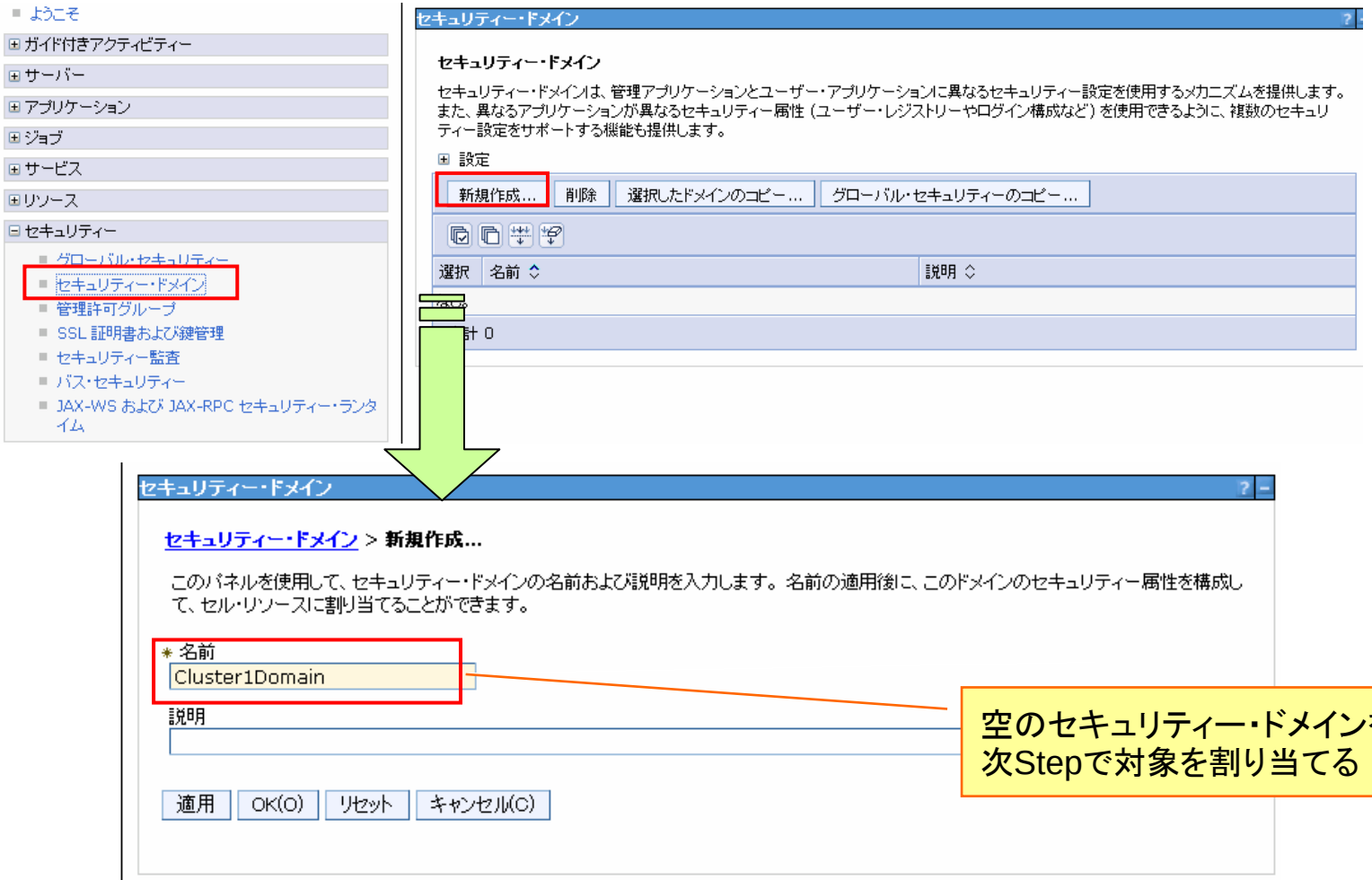


フェデレーテッド（統合）リポジトリーのアーキテクチャー



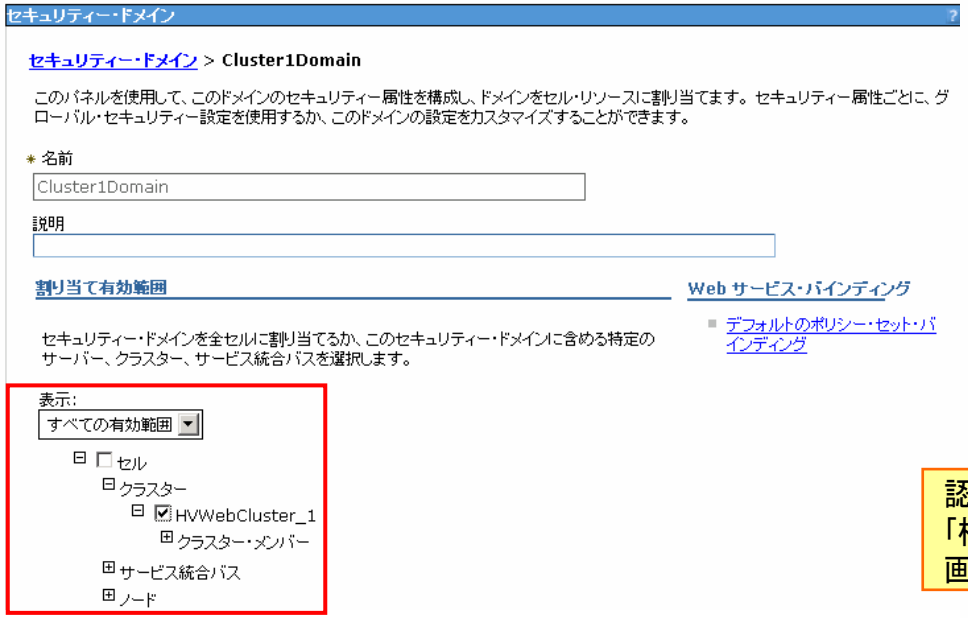
セキュリティー・ドメインの設定方法　(1)

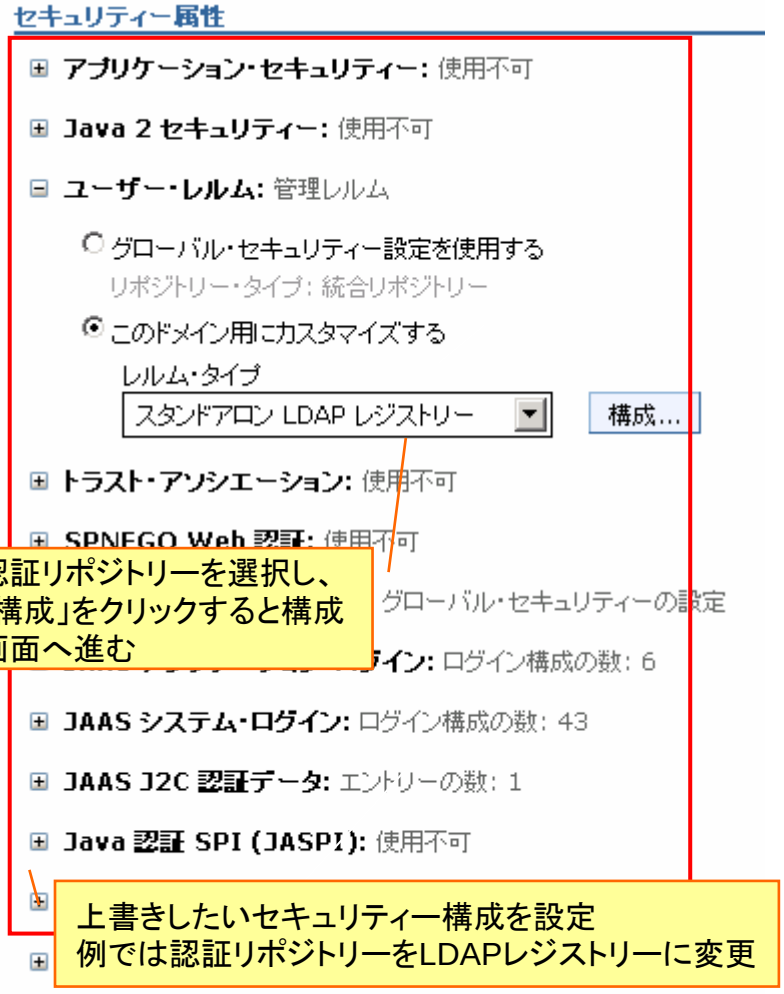
- セキュリティー・ドメインの作成
  - 管理コンソールより、セキュリティー > セキュリティー・ドメインを選択する



セキュリティー・ドメインの設定方法　(2)

- セキュリティー・ドメインの構成
  - 管理コンソールより、セキュリティー > セキュリティー・ドメイン > *[作成したセキュリティー・ドメイン]*を選択する





続き

このセキュリティー・ドメインの割り当て範囲を選択
例ではHVWebCluster_1を選択

認証リポジトリーを選択し、「構成」をクリックすると構成画面へ進む

上書きしたいセキュリティー構成を設定
例では認証リポジトリーをLDAPレジストリーに変更

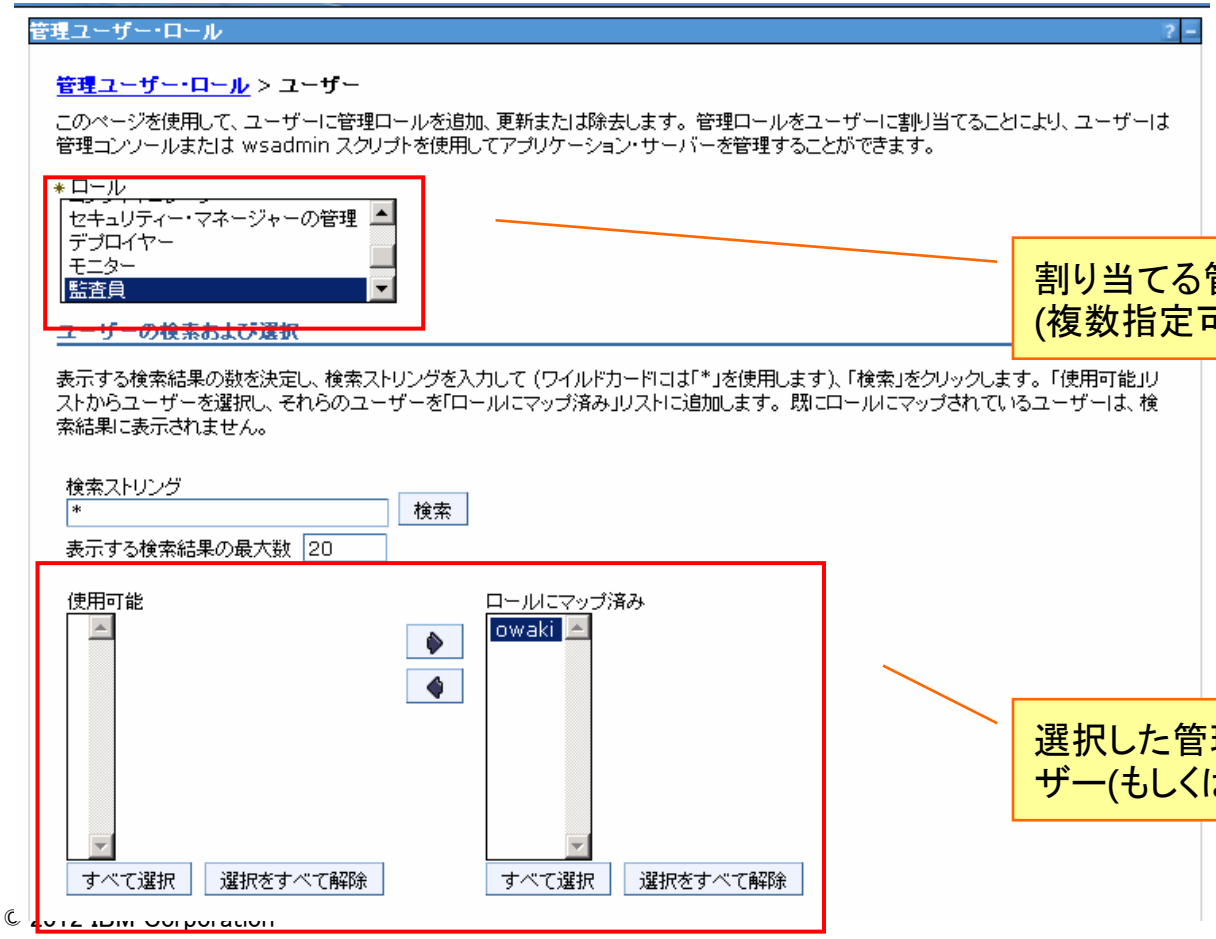
2. WASセキュリティー設計

・セキュリティー技術
・認証
・認可
・暗号化 / 署名
・監査

管理ロールの設定方法

- 管理ロールの設定
  - 管理コンソールより、ユーザーおよびグループ > 管理ユーザー・ロールを選択する

（下記例は管理ユーザー・ロール画面）

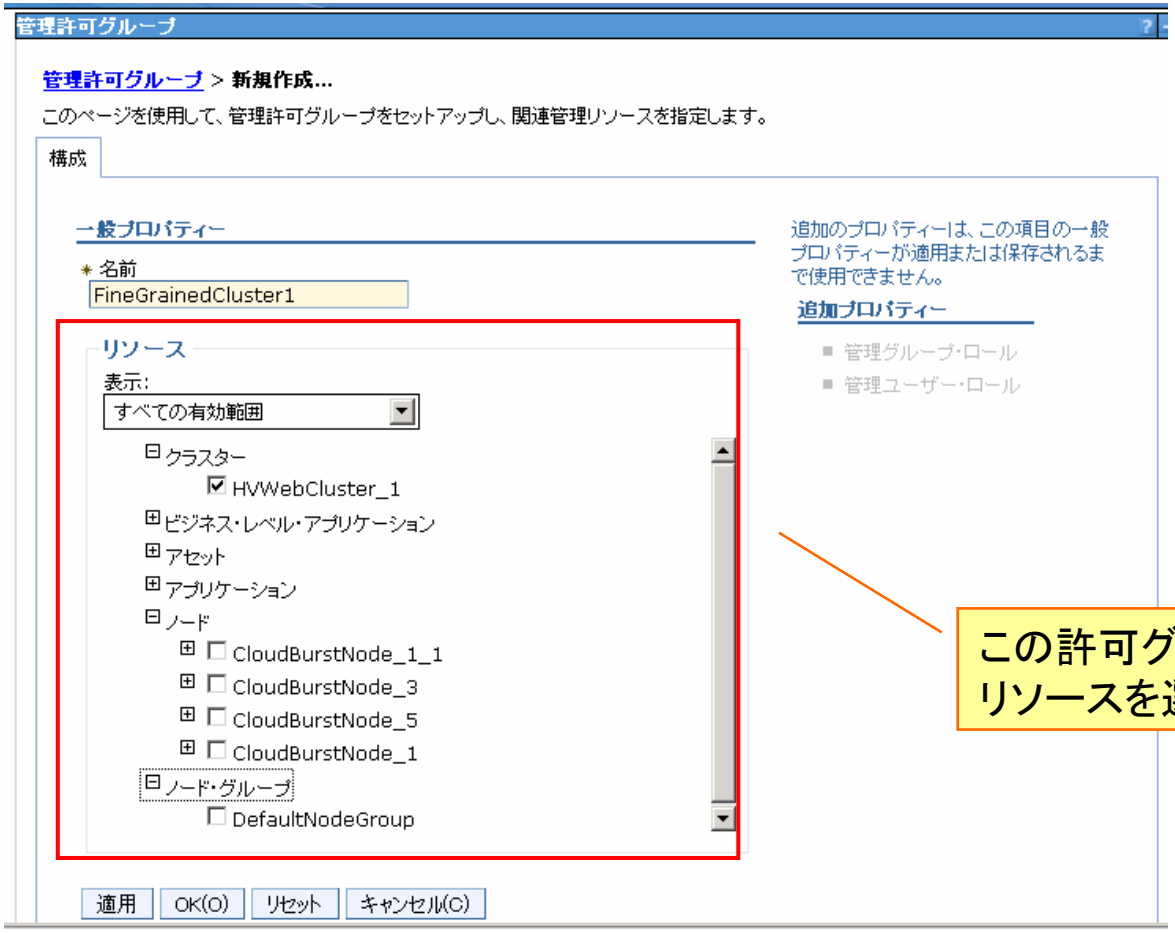


割り当てる管理ロールを選択
(複数指定可)

選択した管理ロールを割り当てるユーザー(もしくはグループ)を指定

詳細な（Fine-grained）管理セキュリティーの設定方法　(1)

- 許可グループを作成
  - 管理コンソールより、セキュリティー > 管理許可グループを選択する

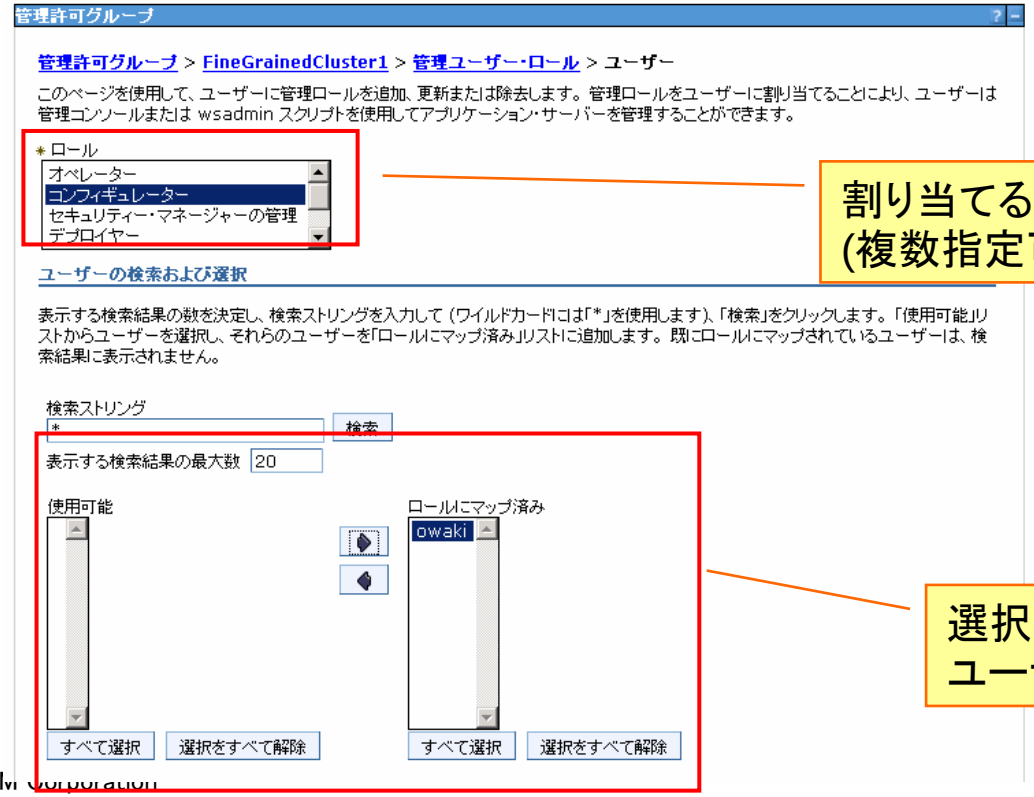


この許可グループに紐付ける
リソースを選択する

詳細な（Fine-grained）管理セキュリティーの設定方法　(2)

- 許可グループを構成
  - 管理コンソールより、セキュリティー > 管理許可グループ > *[作成した許可グループ]* > 管理ユーザー・ロールを選択する
  - 管理コンソールより、セキュリティー > 管理許可グループ > *[作成した許可グループ]* > 管理グループ・ロールを選択する

（例は管理ユーザー・ロール画面）



割り当てる管理ロールを選択
(複数指定可)

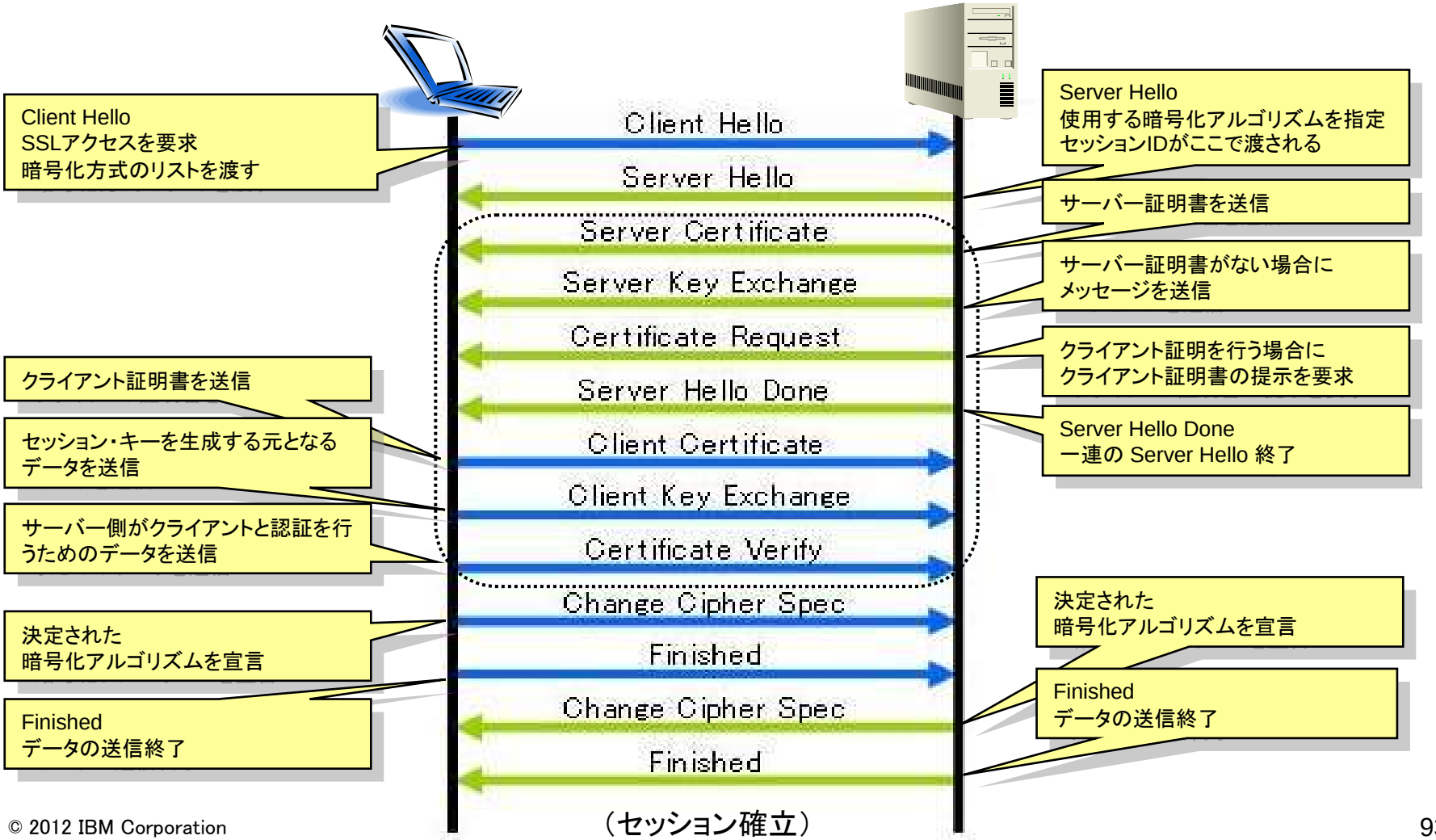
選択した管理ロールを割り当てる
ユーザー(もしくはグループ)を指定

2. WASセキュリティー設計

・セキュリティー技術
・認証
・認可
・暗号化 / 署名
・監査

SSLハンドシェイク

- SSLで通信をする必要があるのか否かを、セキュリティー強度やパフォーマンスの観点から決定する



（セッション確立）

SSLハンドシェイクのキャッシュ

- SSLハンドシェイクは接続に負担がかかるので、二回目からはセッションIDをキャッシュする
  - ErrorLogのレベルをInfo以上にすると、ハンドシェイクのキャッシュが確認可能

error_log (LogLevel info)

```
[Wed Sep 21 13:44:00 2005] [info] Session ID: AABjqsqzy6X02W9ygvVu4Sg9/CpYWFhYQzDlEAAAAAA=
[Wed Sep 21 13:44:00 2005] [info] New Session ID: 1
[Wed Sep 21 13:44:02 2005] [info] Session ID: AABjqsqzy6X02W9ygvVu4Sg9/CpYWFhYQzDlEAAAAAA=
[Wed Sep 21 13:44:02 2005] [info] New Session ID: 0
[Wed Sep 21 13:45:16 2005] [info] Session ID: AABjqsqzy6X02W9ygvVu4Sg9/CpYWFhYQzDlEAAAAAA=
[Wed Sep 21 13:45:16 2005] [info] New Session ID: 0
```

1はSSLの新規接続

0はSSLセッションを再利用

  - SSLV3Timeout、SSLV2Timeoutを設定すると、SSLセッションIDのキャッシュにタイムアウトを設定できる
  - SSLV3はデフォルト120秒、SSLV2はデフォルト40秒

error_log (LogLevel info)

```
[Sun Nov 13 19:37:08 2005] [info] Session ID: AABcXpR4hkJDp2yf9Jln0jQV7sVYWFhYQ3cXVAAAAAA=
[Sun Nov 13 19:37:08 2005] [info] New Session ID: 1
[Sun Nov 13 19:37:10 2005] [info] Session ID: AABcXpR4hkJDp2yf9Jln0jQV7sVYWFhYQ3cXVAAAAAA=
[Sun Nov 13 19:37:10 2005] [info] New Session ID: 0
[Sun Nov 13 19:38:16 2005] [info] Session ID: AABcXpR4h
[Sun Nov 13 19:38:16 2005] [info] New Session ID: 0
[Sun Nov 13 19:38:39 2005] [info] Session ID: AABcXpR4hkJDp2yf9Jln0jQV7sVYWFhYQ3cXVAAAAAA=
[Sun Nov 13 19:38:39 2005] [info] New Session ID: 0
[Sun Nov 13 19:39:23 2005] [info] Session ID: AABcXsIqf40cfw2DV9ILA8lf+dRYWFhYQ3cX2wAAAAE=
[Sun Nov 13 19:39:23 2005] [info] New Session ID: 1
```

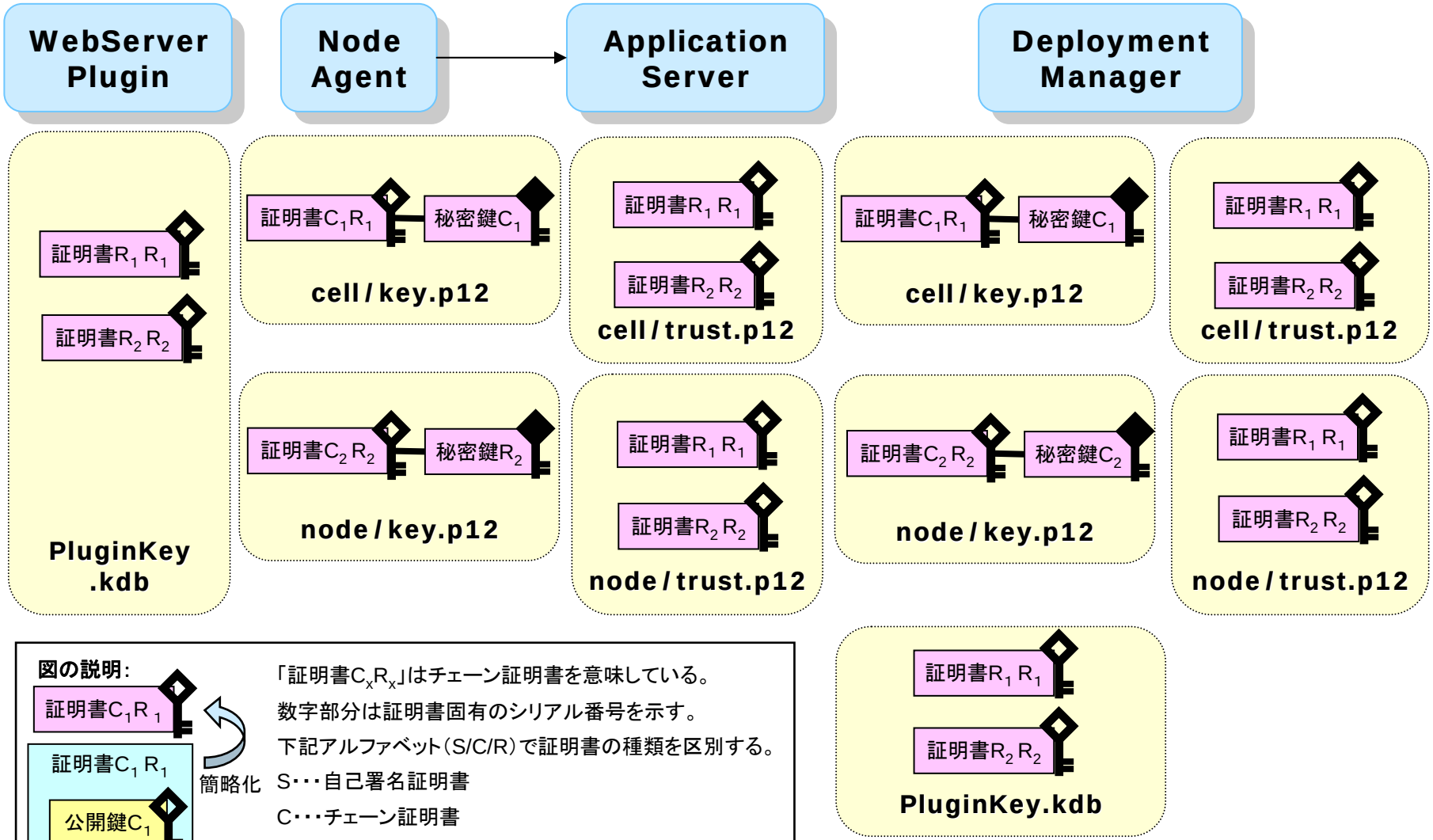
タイムアウトが来るとSSLハンドシェイクをやり直し、新しいセッションIDになる。

WASの各コンポーネント間で使用されるSSL構成

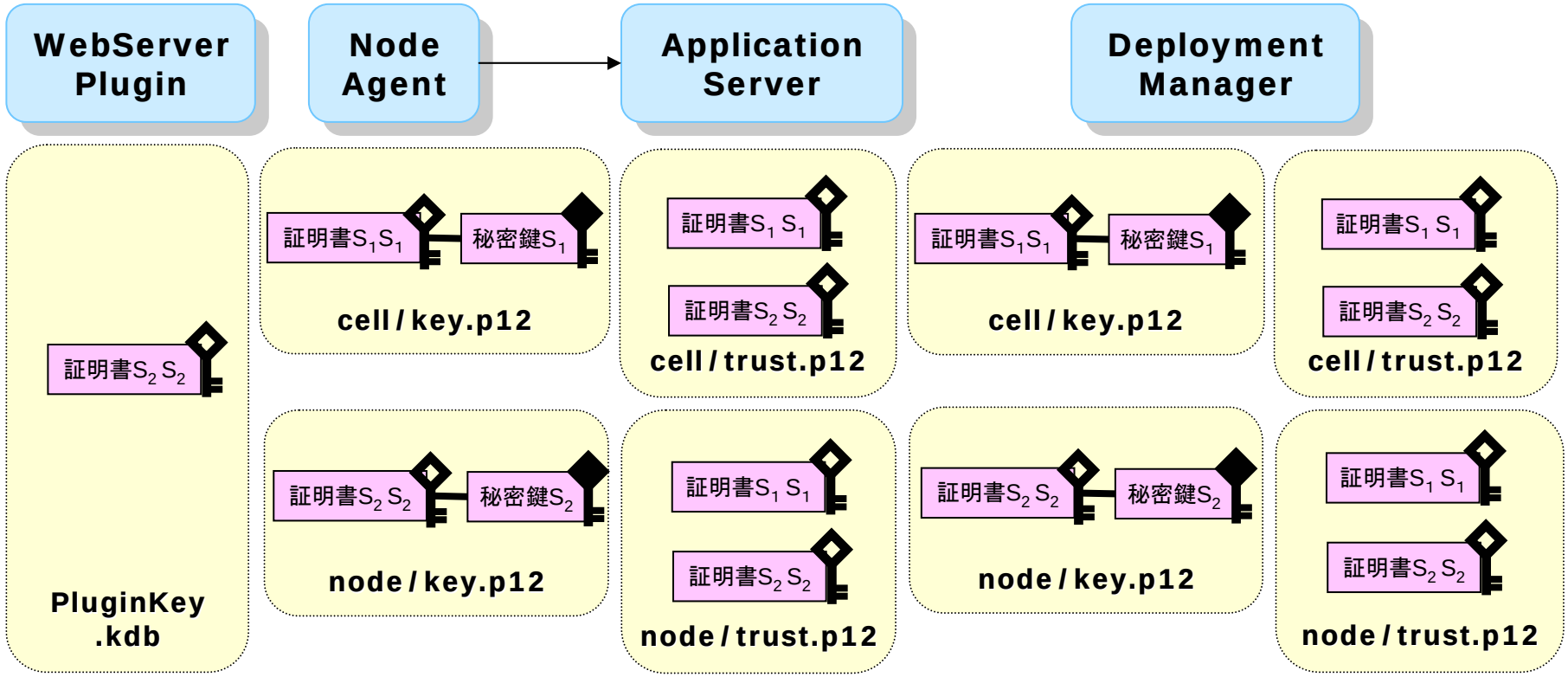
- WASでは、内部にある以下のコンポーネントがSSLを使用しており、データを信頼できるものにし、プライバシーを保護している

- 各コンポーネント間において使用されるSSL
  - HTTPトランスポート
    - WASに組み込まれたHTTPトランスポートは、ブラウザーなどのWebクライアントからSSLを介したHTTP要求を受け入れます
  - ORB
    - WASで使用されるORBは、SSLを介してIIOPを実行し、メッセージを保護します
  - セキュアLDAPクライアント
    - セキュアLDAPクライアントは、SSLを介してLDAPを使用し、LDAPユーザー・レジストリーへのセキュア接続を実現します
    - このクライアントは、LDAPをユーザー・レジストリーとして構成する場合にのみ存在します

WAS V7.0～ 各コンポーネント内の証明書関連図



WAS V6 各コンポーネント内の証明書関連図

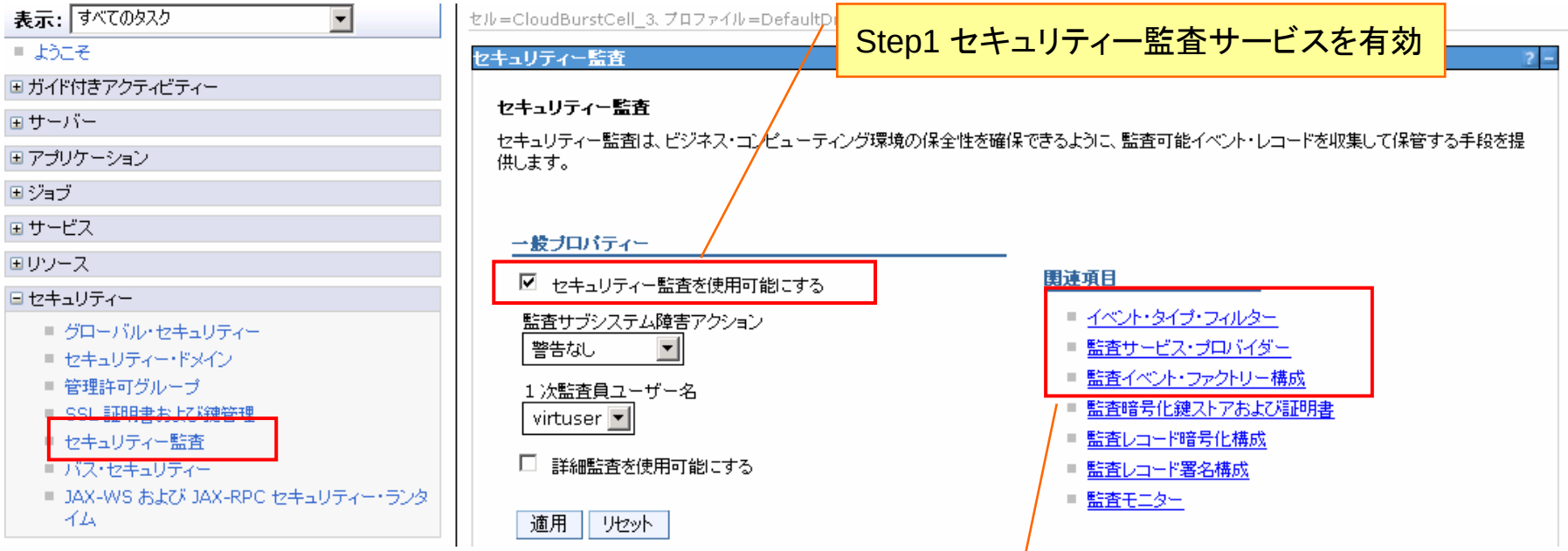


2. WASセキュリティー設計

・セキュリティー技術
・認証
・認可
・暗号化 / 署名
・監査

セキュリティー監査設定方法　(1)

- セキュリティー監査サービスの有効化
  - 管理コンソールより、セキュリティー > セキュリティー監査を選択する

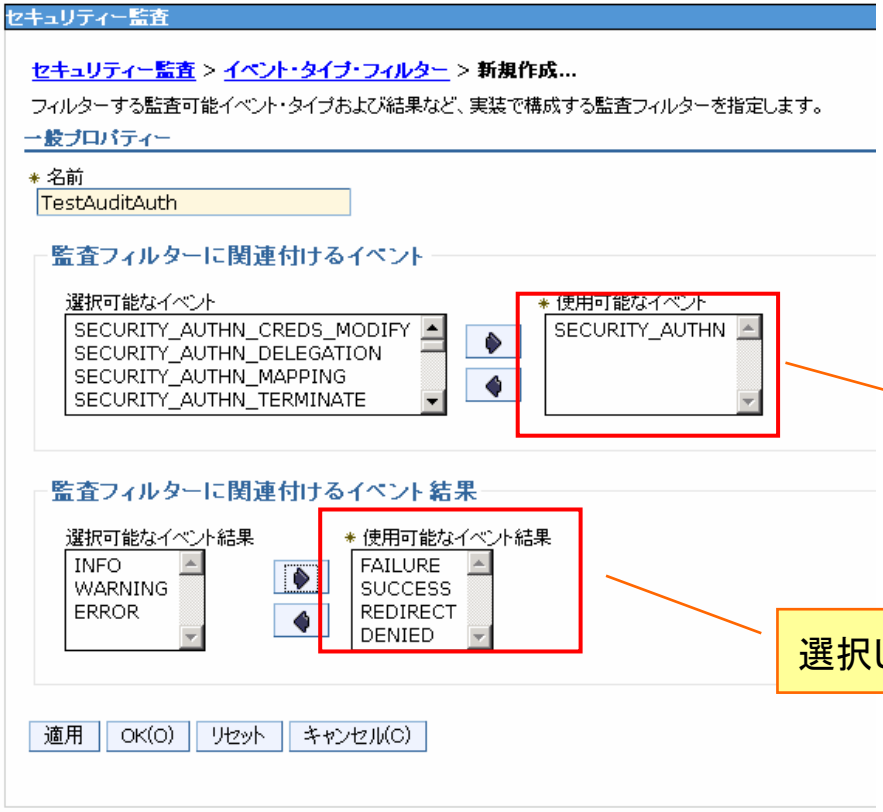


Step1 セキュリティー監査サービスを有効

Step2 イベント・タイプ・フィルターを構成
Step3 監査サービス・プロバイダーを構成
Step4 監査イベント・ファクトリーを構成
詳細は次ページ以降

セキュリティー監査設定方法　(2)

- イベント・タイプ・フィルターの構成
  - 管理コンソールより、セキュリティー > セキュリティー監査 > イベント・タイプ・フィルターを選択する
  - 監査取得対象イベントを定義する

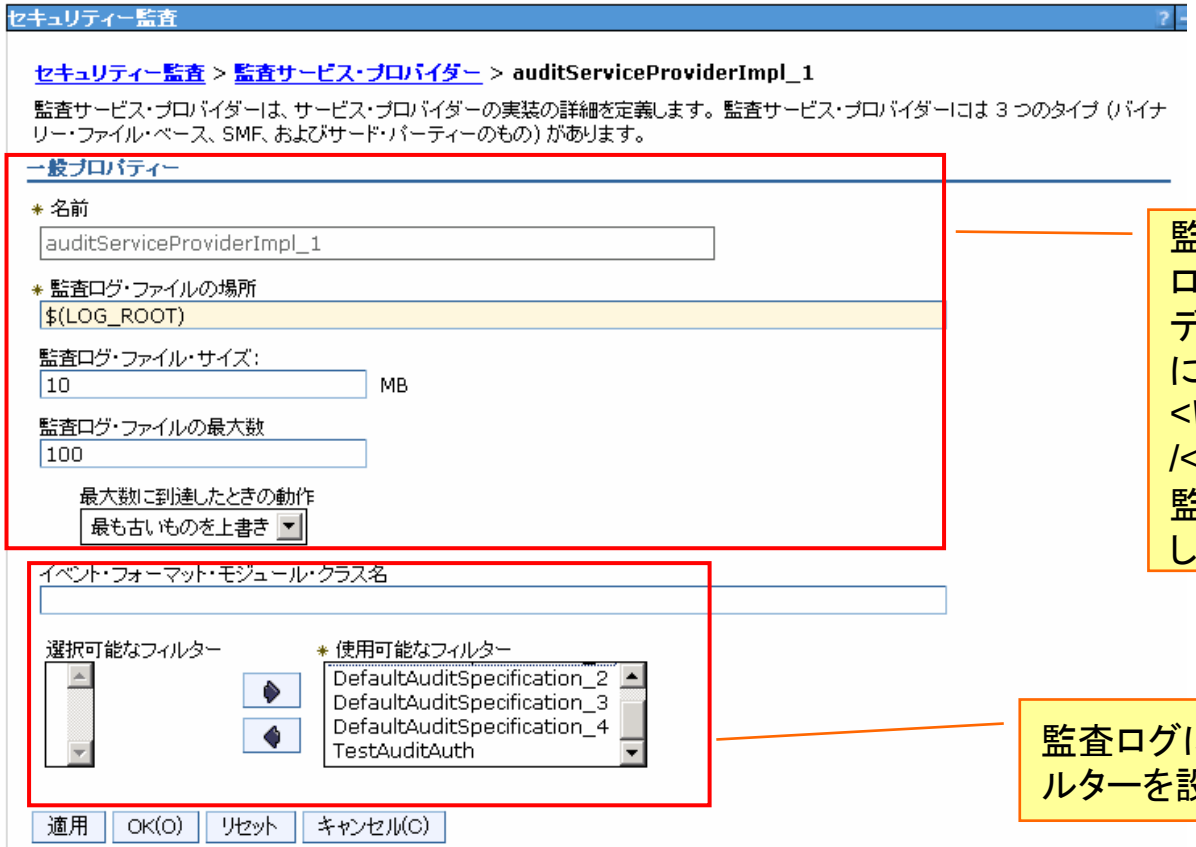


取得対象イベントを選択

選択したイベントに対して、取得したい結果を選択

セキュリティー監査設定方法　(3)

- 監査サービス・プロバイダーの構成
  - 管理コンソールより、セキュリティー > セキュリティー監査 > 監査サービス・プロバイダーを選択する
  - 監査ログ・ファイルの情報を設定する
    - 下記はデフォルトで用意されている監査サービス・プロバイダーをカスタマイズ

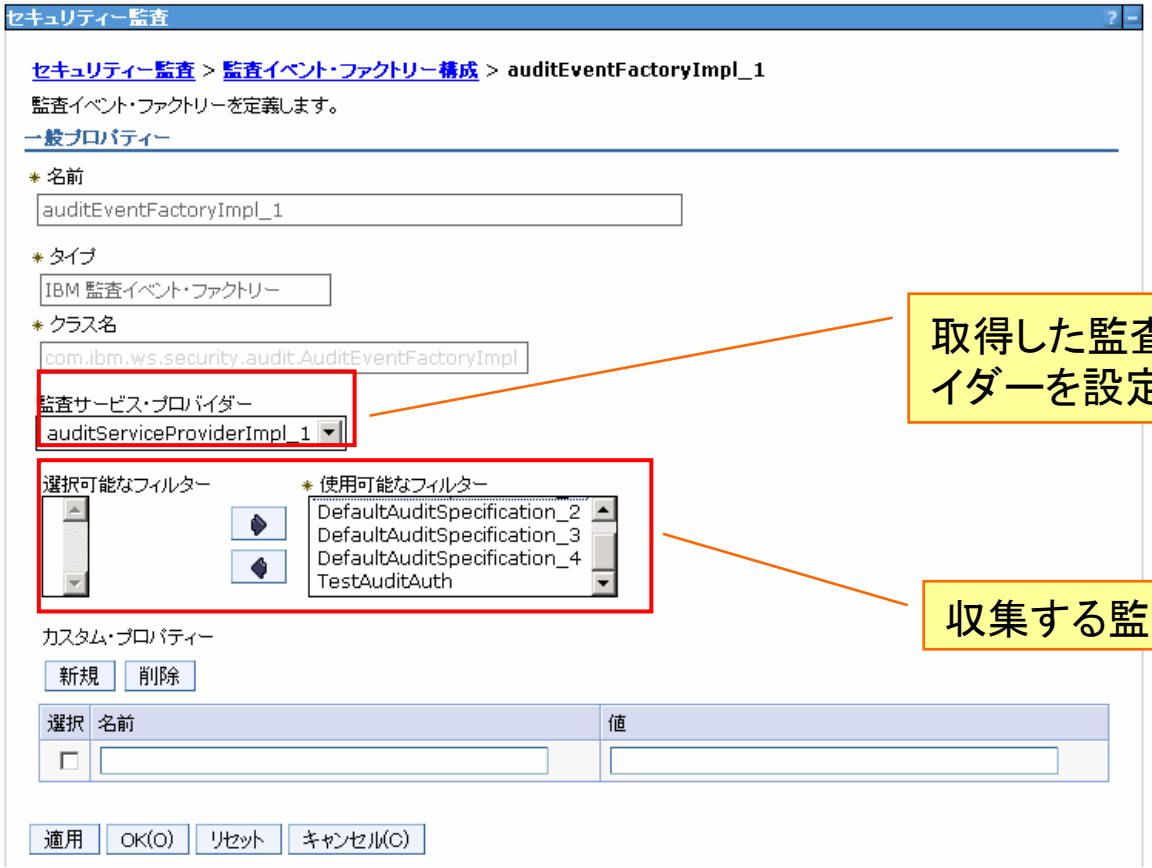


監査ログの出力先ディレクトリー、ログ・ローテーションを設定
デフォルトでは以下ディレクトリーに出力
<*WAS_PROFILE_ROOT*>/logs/<SERVER_NAME>
監査ログ・ファイルの最大数に達したときの動作を指定

監査ログに出力させるイベント・タイプ・フィルターを設定

セキュリティー監査設定方法　(4)

- 監査イベント・ファクトリーを構成
  - 管理コンソールより、セキュリティー > セキュリティー監査 > 監査イベント・ファクトリー構成を選択する
  - 収集する監査データおよび監査データの引き渡し先プロバイダーを設定する
    - 下記はデフォルトで用意されている監査イベント・ファクトリーをカスタマイズ



取得した監査データを引き渡すプロバイダーを設定

収集する監査データを設定

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。 本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものでもありません。 パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、Tivoli、WebSphereは、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。
他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。
現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。

Windowsは Microsoft Corporationの米国およびその他の国における商標です。
JavaおよびすべてのJava関連の商標およびロゴは Oracleやその関連会社の米国およびその他の国における商標または登録商標です。