SiteMinder

Web エージェント設定ガイド

12.52

ca technologies

CA Technologies 製品リファレンス

このマニュアルが参照している CA Technologies の製品は以下のとおりです。

- SiteMinder
- CA Introscope®（旧 CA Wily Introscope）
- CA IdentityMinder™（旧 CA Identity Manager）
- eTrust SOA Security Manager （旧 CA SOA Security Manager）

CA への連絡先

テクニカル サポートの詳細については、弊社テクニカル サポートの Web サイト（http://www.ca.com/jp/support/）をご覧ください。

マニュアルの変更点

SiteMinder の旧リリースで発見された問題の結果として、12.52 のドキュメントに以下の更新が行われました。

- 複数の有効なセッションが存在する場合、レルム タイムアウトの後に再チャレンジを防ぐ（P. 143）-- 新しいパラメータ compatRealmtimeouts を追加しました。（CQ160965、158664、STAR イシュー番号 21025754:01。
- 無効な URL 文字の指定（P. 95）- BadUrlChars のデフォルト値を "デフォルト: 無効（すべての文字を許可）" に修正しました。（CQ: 155294. STAR イシュー： 20933042-1）
- デフォルトの CSS 文字セットのオーバーライド（P. 91）- 内容がより明確になるように書き直されました。（CQ： 155294. STAR イシュー： 20933042-1）
- Web アプリケーション レスポンスの導入（P. 124）- Web アプリケーション クライアント レスポンス機能が基本認証スキームで機能しないという注が追加されました。 CQ 177736 および STAR イシュー 21467829-1 を解決します。
- エラー処理をセットアップする方法（P. 181）- CQ170498、STAR イシュー番号 21389742-01 を解決するために、URL のサンプルを修正しました。
- セッション Cookie の作成または更新の防止（P. 135）- 新規 ACO パラメータを追加しました。
- Cookie が含まれるサーバ レスポンスのキャッシュの防止（P. 366）- CQ171158、CQ171396、STAR イシュー番号 21407131:01 を解決するために、新規 ACO パラメータを追加しました
- 複数の有効なセッションが存在する場合、レルム タイムアウトの後に再チャレンジを防ぐ（P. 143）-- 新しいパラメータ compatRealmtimeouts を追加しました。（CQ160965、158664、STAR イシュー番号 21025754-01）
- HTTP ヘッダ用のレガシー変数の有効化（P. 176）- Apache 2.4.x Web サーバに対する注が追加されました。（CQ178440. STAR イシュー番号 : 21545697）

目次

第 1 章: Web エージェント

このセクションには、以下のトピックが含まれています。

Web エージェントがリソースを保護する方法

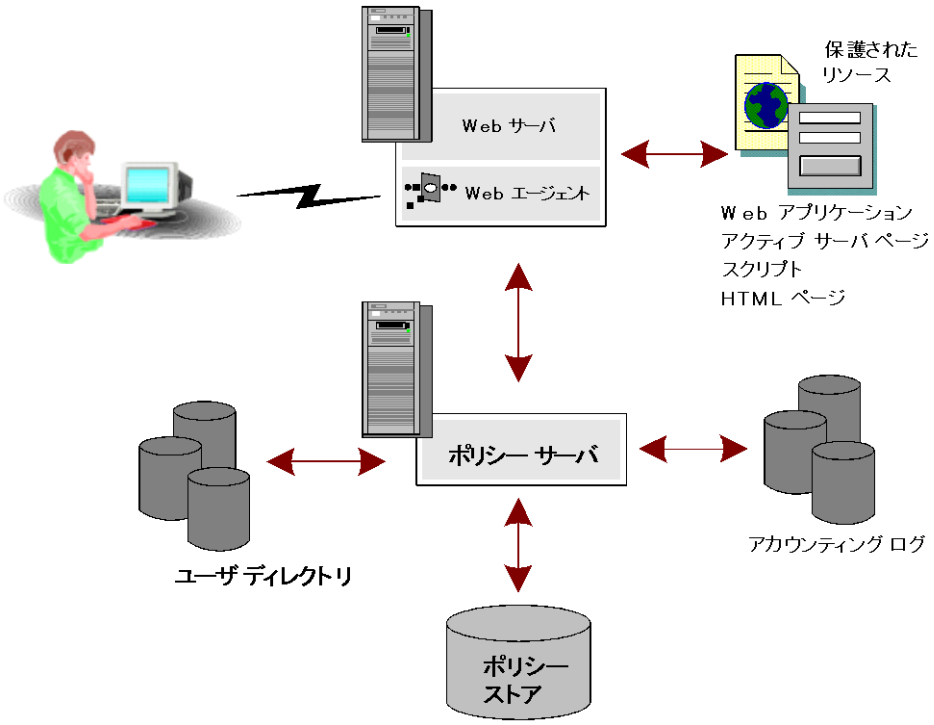
SiteMinder Web エージェントは、URL によって識別できる任意のリソースへのアクセスを制御するソフトウェア コンポーネントです。 Web エージェントは Web サーバに常駐し、リソースの要求をインターセプトして、そのリソースが SiteMinder によって保護されているかどうかを判断します。 その後、Web エージェントはポリシー サーバと連携し、保護された Web サーバ リソースへのアクセスを要求するユーザの認証および許可を行います。

Web エージェントは以下のタスクを実行します。

- 保護されているリソースへのアクセスリクエストをインターセプトし、ポリシー サーバと連携して動作し、ユーザがアクセス権を持っているかどうかを判断します。
- ユーザに対するコンテンツの提示方法（ポリシー ベースのパーソナライゼーション）とアクセス権限の配信方法を指示する Web アプリケーションに情報を提供します。
- ユーザが情報に迅速かつ安全にアクセスできるようにします。 Web エージェントはユーザ アクセス権限に関するコンテキスト情報をセッション キャッシュに格納します。キャッシュ設定値を変更すると、パフォーマンスを最適化できます。
- 単一の cookie ドメインまたは複数の cookie ドメインにある複数の Web サーバ間でシングル サインオンを有効にして、ユーザの再認証を不要にします。

SiteMinder Web エージェントおよびサポートされている Web サーバ プラットフォームの一覧については、「テクニカル サポート」にアクセスし、SiteMinder サポート マトリックスを検索してください。

Web エージェントは、以下の図に示すように、Web サーバ上にあります。


保護された
リソース
Web サーバ
Web エージェント
Web アプリケーション
アクティブ サーバ ページ
スクリプト
HTML ページ
ポリシー サーバ
アカウンティング ログ
ユーザ ディレクトリ
ポリシー
ストア

Web エージェントとポリシー サーバが連携する仕組み

アクセス制御を実行する場合、Web エージェントは、ポリシー サーバと対話を行います。実際に許可と認証の判断を行うのは、ポリシー サーバです。

Web エージェントはリソースに対するすべてのユーザ リクエストをインターセプトし、要求されたリソースが保護されているかどうかをポリシー サーバに確認します。 リソースが保護されていない場合は、アクセス要求は Web サーバに直接渡されます。 リソースが保護されている場合、次の動作が発生します。

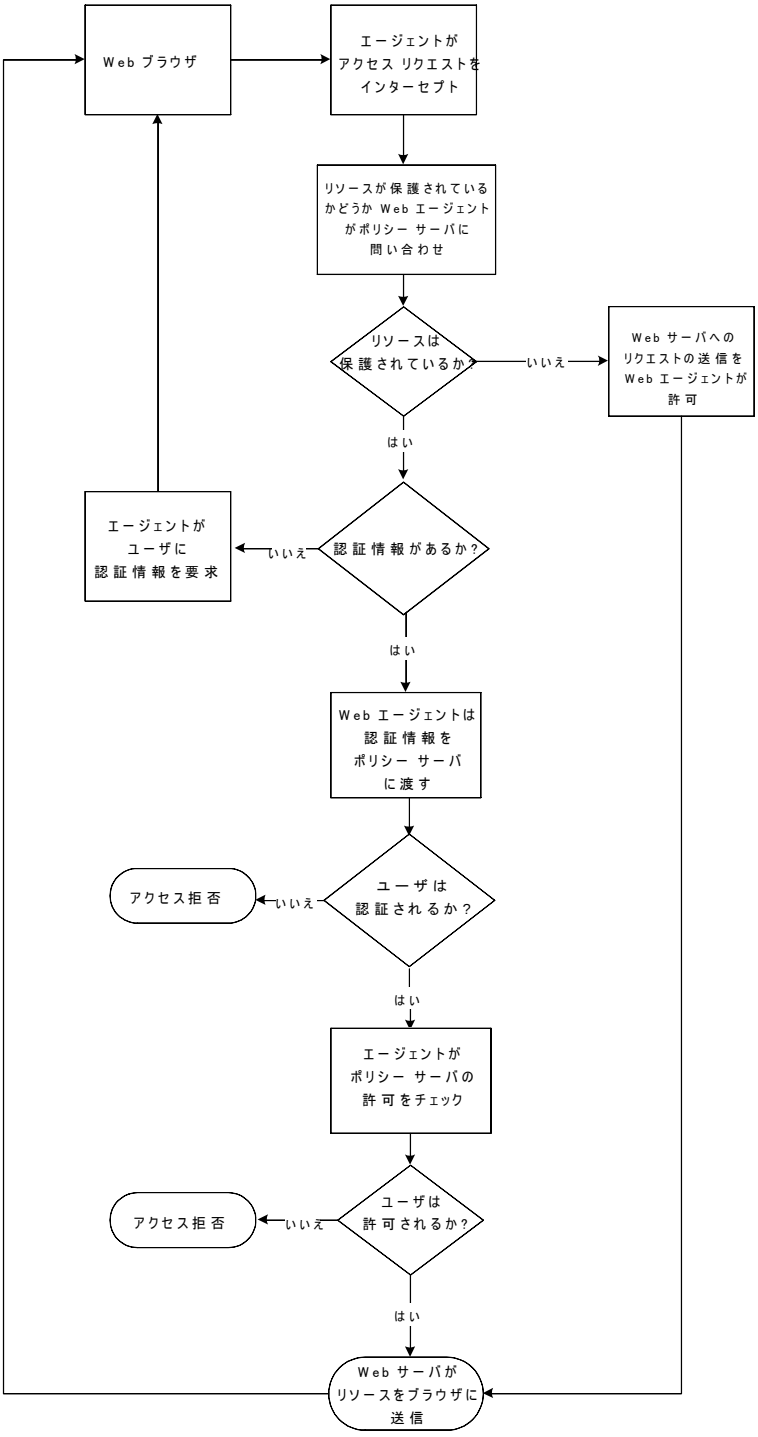
1. Web エージェントは、そのリソースに関して、どの認証方法が必要とされているのかチェックします。 一般的な認証情報は名前とパスワードです。しかし、証明書、トークンカードの PIN （個人用識別番号）のような他の認証情報が必要になる場合もあります。
2. Web エージェントは、認証情報の入力をユーザに要求します。

   ユーザはそれに応答し、適切な認証情報をレスポンスとして返します。
3. Web エージェントは、これらの認証情報をポリシー サーバに渡します。ポリシー サーバは、その認証情報が正しいかどうかを判断します。
4. ユーザが認証フェーズに合格した場合、ポリシー サーバは、ユーザがそのリソースにアクセスすることを許可されているかどうか判断します。ポリシー サーバがアクセスを許可した後、Web エージェントは、その要求を Web サーバに渡します。

Web エージェントは、ユーザ固有の属性も *レスポンス*の形式で受信します。これにより、Web コンテンツのパーソナライズ機能とセッション管理が可能になります。レスポンスは、ユーザの許可後にポリシー サーバから Web エージェントに返されるパーソナライズされたメッセージやユーザ固有の情報です。レスポンスは、名前/値という属性ペアで構成されています。この属性ペアは、Web アプリケーションで使用するために、Web エージェントが HTTP ヘッダに追加したものです。 レスポンスの例には、以下のようなものがあります。

- Web エージェントは、Web アプリケーションへのアクセスをユーザに許可した後、ユーザ セッションの持続時間を指示する情報も Web アプリケーションに送信できます。
- また、以前に登録したサイトに再びアクセスした場合に、Web エージェントはユーザの購買志向に関する情報を返すこともできます。

以下の図は、Web エージェントとポリシー サーバの間の通信を示しています。



異なるタイムゾーンにある Web エージェントとポリシー サーバについての考慮事項

デフォルトでは、ポリシー サーバと Web エージェントは GMT（グリニッジ標準時）を基準にして時間を計算します。 したがって、ポリシー サーバまたは Web エージェントがインストールされている各システムのシステム クロックを、その地域のタイム ゾーンに基づいて設定しておく必要があります。

以下の図に、ポリシー サーバが時間を基準にポリシーをどのように実行するかを示します。 リソースは、マサチューセッツにある Web サーバに格納されていて、カリフォルニアにあるポリシー サーバで保護されています。 このポリシーでは、午前 9 時から午後 5 時までの間のリソースへのアクセスを許可します。ただし、マサチューセッツのユーザは午後 6 時でもリソースにアクセスできます。これは、このポリシーがポリシー サーバのタイムゾーンである PST（太平洋標準時）に基づいており、PST は Web エージェントのタイムゾーンである EST（東部標準時）よりも 3 時間遅れているためです。

ポリシー サーバの地域はまだ午後 3 時であるため、マサチューセッツ州のユーザは午後 6 時でもリソースにアクセスできます。

注: Windows システムでは、タイムゾーンと時刻（［日付と時刻］コントロール パネルで設定）の両方が整合している必要があります。たとえば、米国でシステムを東部標準時から太平洋標準時にリセットするには、以下の順にタスクを実行します。

a. タイムゾーンを太平洋標準時に設定します。

b. システム クロックに正しい時間（東部標準時より 3 時間早い時間）が表示されていることを確認します。

これらの設定値が整合していない場合、複数のドメイン間でのシングル サインオンや、エージェント キー管理機能が正しく動作しません。

エージェントが SiteMinder の cookie を読み取る方法

Web エージェントは、エージェント キーを使用して、SiteMinder の cookie の暗号化と復号化を行い、cookie に格納されているデータを読み取ることができるようにします。 エージェントはそのキーを使用して cookie を暗号化した後、その暗号化済み cookie をユーザのブラウザへ送信し、他の Web エージェントから受信した cookie を復号化します。

すべての Web エージェントが同じキーを知っている必要があります。また、1 つのポリシー サーバと通信を行うすべてのエージェントのキーを、同じ値に設定する必要があります。 このルールは、シングル サインオン環境にあるエージェントの場合、特に重要です。 キーの安全を確保するため、ポリシー サーバはキーの「*ロール オーバー*」を実行します。 キーのロールオーバーとは、新しいキーを生成して暗号化し、SiteMinder 環境内のすべての Web エージェントにそれらのキーを配布することです。

Web エージェントが起動し、管理呼び出し(リクエスト)を行った時点で、ポリシー サーバは現在のキーセットを提供します。Web エージェントは、ポリシー サーバをポーリングするたびに、管理呼び出しを繰り返します。Web エージェントは、更新済みのキーを受け取ります。

ポリシー サーバは、以下のタイプのキーを提供します。

ダイナミック キー

ポリシー サーバのアルゴリズムにより生成され、接続された他のポリシー サーバや関連する Web エージェントに配布されるキーです。 ダイナミック キーは、一定の間隔で自動的にロールオーバーできます。また、管理 UI を使用して手動で変更することもできます。

スタティック キー

常に同一であるキーです。ポリシー サーバのアルゴリズムによって生成するか、手動で設定することができます。SiteMinder では、cookie に情報を長期間保存する必要のある機能のサブセットに、このタイプのキーを使用します。

自動キー変換を使用すると、1 つの キー ストアを共有する大規模な SiteMinder インストール環境でエージェント キーの管理プロセスを簡易化することができます。 キー ストアとは、すべてのキー情報のストレージ ロケーションです。 ポリシー サーバは、このキー ストアにアクセスして現在のキーを取得し、そのキーが Web エージェントに渡されます。 シングル サインオンを設定されたエージェントの場合、キーストアを複製して、シングル サインオン環境のすべてのポリシー サーバでキーストアを共有する必要があります。 自動キー変換により、キーの完全性も確保されます。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

Web エージェントとダイナミック キーのロールオーバー

管理 UI を使用して、ダイナミック エージェント キーのロールオーバーを設定することができます。 Web エージェントは、キーの更新があるかどうかポリシー サーバを定期的にポーリングします。 キーが更新されている場合、Web エージェントはポーリング時に変更内容を取得します。 デフォルトのポーリング時間は 30 秒ですが、この値は、Web エージェントの PSPollInterval パラメータの値を変更することで、カスタマイズできます。

Web エージェントは、キーのロールオーバーが発生したことを検出した時点で、以下のエージェントキーの新しい値を取り出します。

前回キー

現在の値の前にダイナミック エージェント キーに使用していた最後の値が入ります。

現在キー

現在のダイナミック エージェント キーの値が入ります。

予定キー

ダイナミック エージェント キーのロールオーバーで*現在*キーとして使用する次回の値が入ります。

スタティック キー

エージェントが、ユーザを識別してその情報を長期間保存する必要がある SiteMinder 機能に使用できる、長期間キーが入ります。 ダイナミックキーが使用できない場合、スタティックキーは、シングル サインオンに関連して cookie の暗号化もサポートします。

Web エージェントでは、cookie データを保持したり、古いキーから新しいキーへスムーズに移行するために、複数のキーが必要です。

詳細情報:

エージェントがポリシーまたはキーの更新をチェックする頻度の変更 (P. 82)

キーストア

ポリシー サーバは、生成したダイナミック キーを、キー ストアに保存して管理します。 キー ストアはリポジトリであり、すべてのポリシー サーバは最新のキーをここから取得します。 Web エージェントは、ポリシー サーバから現在キーを取得します。キー ストアは、SiteMinder ポリシー ストアの一部に組み込むことも、スタンドアロン キー ストアとして保持することもできます。

注: 管理者が、エージェント キーの複数のロールオーバーを短時間に続けて実行した場合、このアクションにより、シングル サインオン用のすべての cookie が無効になり、現在ログイン中のすべてのユーザのシングル サインオンが無効になる可能性があります。 これらのユーザが再認証されると、シングル サインオンは正常に動作するようになります。

フレームワークと従来のエージェント アーキテクチャ

すべての SiteMinder エージェントは、次の*いずれか*のアーキテクチャに基づいています。

- 従来のエージェントは、オリジナルの SiteMinder エージェント アーキテクチャに基づいています。
- フレームワーク エージェントは、SiteMinder バージョン 5.x QMR 6 で導入されました。

エージェント機能は、アーキテクチャにかかわらず基本的に同じですが、いくつかの小さな違いがあります。 たとえば、フレームワーク エージェントは別の WebAgent.conf ファイルおよび LocalConfig.conf ファイルを使用します。 従来のエージェントはこれらのファイルを使用*しません*。

従来のエージェントは、次の Web サーバにインストールされます。

- Domino （サポートされているすべてのバージョン）

フレームワーク エージェントは、以下の Web サーバにインストールされます。

- Microsoft Internet Information Services （IIS） 7.0、7.5
- Apache 2.0.54、2.2.x および他の Apache 2.0 ベースのサーバ（IBM HTTP Server および HP Apache サーバなど）
- Oracle iPlanet Web サーバのバージョン 6.1 以降

  注: Oracle iPlanet Web Server は以前「Sun Java Systems」または「SunONE」という名前でした。

詳細情報:

フレームワーク エージェントと従来のエージェントの間の POST 維持の有効化 (P. 205)

変更時にサーバの再起動を必要とするパラメータ

一部のエージェント パラメータは、動的に更新されます。以下のパラメータに変更を適用した場合は、Web サーバを再起動する必要があります。

AgentConfigObject

ローカル エージェント設定ファイル内にエージェント設定オブジェクト（ポリシー サーバに格納された）の名前を定義します。このパラメータはエージェント設定オブジェクトでは*使用されません*。

デフォルト： デフォルトなし

CacheAnonymous

Web エージェントが匿名のユーザ情報をキャッシュするかどうかを指定します。 このパラメータは、たとえば以下の状況に対して設定できます。

- Web サイトのユーザのほとんどが匿名ユーザで、それらのユーザのセッション情報を格納したい場合。
- 登録ユーザと匿名ユーザの両方が Web サイトにアクセスする場合。

  匿名ユーザの情報のみでキャッシュが満杯になり、登録ユーザ用の領域がなくなってしまう可能性がある場合は、このパラメータを無効にすることをお勧めします。

デフォルト： No

HostConfigFile

トラステッド ホスト コンピュータがポリシー サーバに正常に登録された後に作成される SMHost.conf ファイル（IIS 6.0 または Apache のエージェント内）のパスを指定します。 コンピュータ上のすべての Web エージェントが SMHost.conf ファイルを共有します。

デフォルト： デフォルトなし

MaxResourceCacheSize

Web エージェントがそのリソース キャッシュ内で保持するエントリの最大数を指定します。エントリには以下の情報が含まれます。

- リソースが保護されるかどうかに関するポリシー サーバのレスポンス

- レスポンスで返される追加属性

最大値に達すると、新しいリソース レコードが最も古いリソース レコードと置き換わります。

これらをより大きな数値に設定する場合は、十分なシステム メモリがあることを確認してください。

OneView モニタを使用して Web エージェント統計を表示している場合は、ResourceCacheCount に表示される値が MaxResourceCacheSize パラメータで指定された値より大きいことがあります。これはエラーではありません。Web エージェントは、MaxResourceCacheSize パラメータを 1 つのガイドラインとして使用します。また、値は状況により異なります。これは、MaxResourceCacheSize パラメータはリソース キャッシュ内の平均サイズのエントリの最大数を示すためです。 実際のキャッシュ エントリは、あらかじめ識別された平均サイズより大きかったり小さかったりする可能性があります。したがって、実際の最大エントリ数は指定された値より多い場合や少ない場合があります。

注: フレームワーク エージェントなど、共有メモリを使用する Web エージェントの場合、キャッシュは MaxResourceCacheSize の値に基づいて一定サイズが事前に割り当てられ、それより増えることはありません。

デフォルト： （Domino Web サーバ） 1000

デフォルト： （IIS および Sun Java System Web サーバ） 700

デフォルト： （Apache Web サーバ） 750

MaxSessionCacheSize

エージェントがそのセッション キャッシュ内で保持するユーザの最大数を指定します。 セッション キャッシュには、認証するユーザのセッション ID が正常に格納されます。 それらのユーザが同じセッション中に同じレルム内の別のリソースにアクセスした場合、エージェントはポリシー サーバをコールする代わりにセッション キャッシュの情報を使用します。この最大数に達すると、エージェントは最も古いユーザ レコードを新しいユーザ レコードと置き換えます。

このパラメータの値は、持続期間にリソースにアクセスしてそれを使用する予定のユーザの数に基づいて設定します。 これらをより大きな数値に設定する場合は、十分なシステム メモリがあることを確認してください。

デフォルト： （Domino Web サーバ） 1000

デフォルト：（IIS および Oracle iPlanet Web サーバ） 700

デフォルト：（Apache Web サーバ） 750

PostPreservationFile

以下の POST 維持テンプレート ファイルの*いずれか*に対するパスを指定することで、トラディショナル エージェントとフレームワーク エージェントとの間の POST 維持データの転送を有効にします。

- tr2fw.pptemplate - トラディショナル エージェントが稼働しているサーバでホストされているリソースが、フレームワーク エージェント上で実行されている FCC によって保護されていることを示します。
- fw2tr.pptemplate - フレームワーク エージェントが稼働しているサーバでホストされているリソースが、トラディショナル エージェント上で実行されている FCC によって保護されていることを示します。

デフォルト： デフォルトなし

例： *web_agent_home*/samples/forms/fw2tr.pptemplate

ResourceCacheTimeout

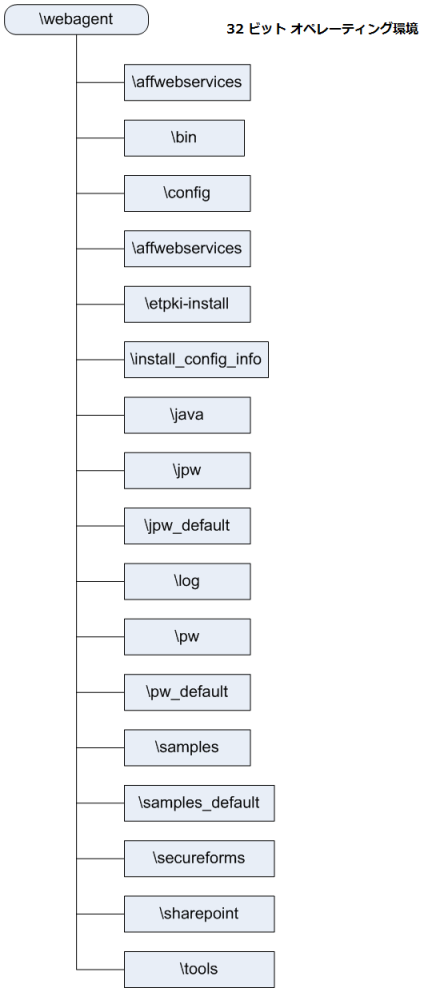
リソース エントリがキャッシュに保存される秒数を指定します。時間間隔の値を超えると、Web エージェントはキャッシュされたエントリを削除します。その後、保護されているリソースにユーザがアクセスしようとすると、Web エージェントはポリシー サーバに問い合わせます。

デフォルト： 600（10 分）

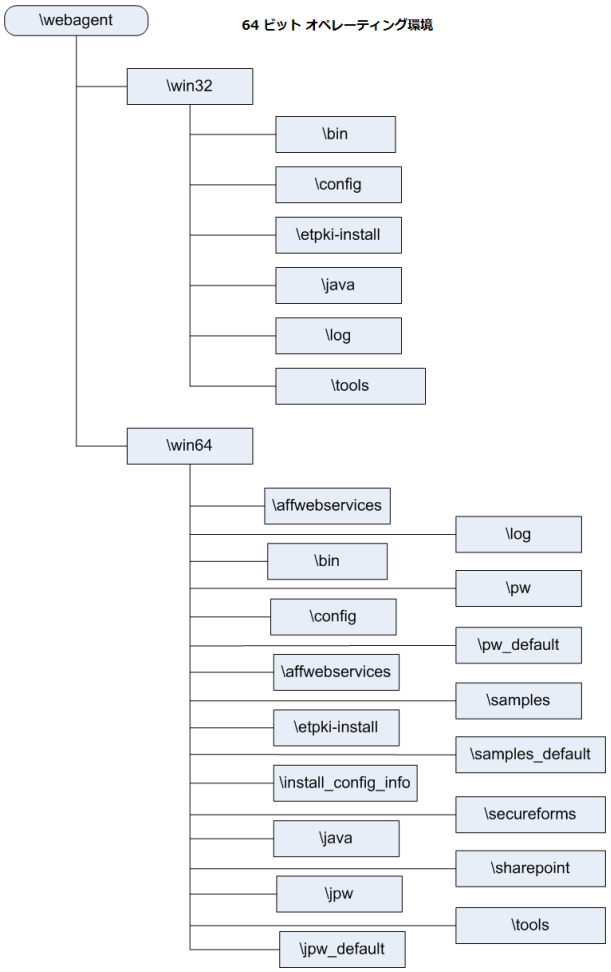
オペレーティング環境に従った IIS ディレクトリ構造用の複数エージェント

エージェント ファイルの IIS Web サーバに追加されたディレクトリ構造は、IIS Web サーバのオペレーティング環境に応じて異なります。 以下のディレクトリ構造があります。

- SiteMinder Web エージェントおよび IIS 用の [set AGENT value for your book] は、以下の図に示すディレクトリ構造を使用します。



- 64 ビット オペレーティング環境にインストールされた IIS の SiteMinder エージェントは、以下の図に示すディレクトリ構造を使用します。



第 2 章: エージェントの設定方法

このセクションには、以下のトピックが含まれています。

中央設定

中央エージェント設定では、ポリシー サーバのエージェント設定オブジェクトから 1 つ以上の Web エージェントを管理します。 ポリシー サーバにあるエージェント設定オブジェクトには、Web エージェントが使用するパラメータが入っています。中央設定の 1 つの利点は、複数のエージェントのパラメータ設定を同時に更新できることです。 ほとんどのパラメータ変更は動的に発生しますが、フレームワークのパラメータの中には、変更後に Web サーバの再起動を必要とするものもあります。

エージェント設定オブジェクトの作成および編集には、管理 UI を使用します。 ポリシー サーバと通信するそれぞれの Web エージェントは、エージェント設定オブジェクトに関連付ける必要がありますが、複数の Web エージェントで 1 つのエージェント設定オブジェクトを使用することができます。

注: エージェント設定オブジェクトの作成方法の詳細については、ポリシー サーバのマニュアルを参照してください。

詳細情報

変更時にサーバの再起動を必要とするパラメータ (P. 27)

中央設定の実装

中央設定はデフォルトで有効になっています。 エージェントでは、設定ウィザードでエージェントを設定したときに指定した既存のエージェント設定オブジェクトの構成設定が使用されます。 パラメータの設定は、必要に応じていつでも変更できます。

次の手順に従ってください:

1. 管理 UI にログインします。

   ［Welcome］画面が表示されます。

2. ［インフラストラクチャ］-［エージェント設定オブジェクト］をクリックします。

   エージェント設定オブジェクトのリストが表示されます。

3. 目的の［エージェント設定オブジェクト］の行で、変更アイコンをクリックします。

   ［エージェント設定の変更］ウィンドウが表示されます。

4. AllowLocalConfig パラメータの値が no に設定されていることを確認します。

5. 必要に応じて、管理 UI を使用して他のパラメータの設定を変更します。

6. ［サブミット］をクリックします。

   ［エージェント設定の変更］ウィンドウが閉じ、確認メッセージが表示されます。

7. （オプション）将来の参照用に、変更に関するコメントを［コメント］フィールドに入力します。

8. ［はい］をクリックします。

   確認メッセージが表示されます。中央設定が実装されます。ほとんどのパラメータは動的に変更されますが、一部の変更を有効にするには Web サーバの再起動が必要です。

詳細情報:

変更時にサーバの再起動を必要とするパラメータ (P. 27)

ローカル エージェント設定

ローカル設定

ローカル エージェント設定では、Web サーバをホストしているシステム上にインストールされているローカル ファイルを使用して、Web エージェントを管理します。ローカル ファイル内のパラメータ設定は、ポリシー サーバ上のエージェント設定オブジェクトに格納されているすべての設定に優先します。エージェント設定オブジェクト内の設定は変更されません。ローカル エージェント設定を考慮する状況には以下があります。

- たとえば、Apache Web エージェントが 3 つあり、最初の 2 つ（A と B）は同一のパラメータ設定を使用しており、3 つ目の Apache エージェント（C）は、A と B の設定の大半を使用する一方で、リバース プロキシとして動作している場合。これを実行するには、Apache エージェント A および B のセントラル エージェント設定を使用し、Apache エージェント C にはローカル設定を使用します。

- ポリシー サーバ管理者がエージェントを設定する同一人物（またはグループ）でない場合。たとえば、社内の IT 部がポリシー サーバをメンテナンスしているけれども、財務部がエージェントを使用して会計アプリケーションへのアクセスを制御している場合。IT 部の担当者は、ポリシー サーバ上でエージェントのローカル設定を実行できるのに対し、財務部の担当者は、会計アプリケーションを保護するエージェントの特定の設定を制御します。

フレームワーク Web エージェントでは、ローカル設定に以下のファイルを使用します。

WebAgent.conf

ポリシー サーバの起動および接続にフレームワーク Web エージェントが使用する中心的な設定が含まれます。

LocalConfig.conf

フレームワーク Web エージェントの設定が含まれます。

トラディショナル Web エージェントでは、ローカル設定に以下のファイルを使用します。

WebAgent.conf

従来の Web エージェントの設定がすべて含まれます。

WebAgent.conf ファイルのロケーション

以下の表に、各種の Web サーバでの WebAgent.conf ファイルの場所を示します。

web_agent_home

SiteMinder エージェントがインストールされているディレクトリを示します。

デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）： C:¥Program Files¥CA¥webagent

デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）： C:¥Program Files¥CA¥webagent¥win64

デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）： C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）： /opt/ca/webagent

Web サーバ	ファイルの場所
IIS	*web_agent_home*¥bin¥IIS
Oracle iPlanet （iPlanet/SunOne）	*Oracle_iPlanet_server_home*/https-*hostname*/config ここで、*Oracle_iPlanet_home* は、Oracle iPlanet Web サーバのインストール場所で、*hostname* はサーバの名前です。
Apache IBM HTTP Server Oracle HTTP Server	*web_server_home*/conf ここで、*web_server_home* は、Web サーバのインストール場所です。
Domino	Windows： c:¥lotus¥domino UNIX： $HOME/notesdata

詳細情報:

Web エージェントの有効化 (P. 77)
Web エージェントの無効化 (P. 78)

フレームワーク エージェントの WebAgent.conf ファイル

AgentConfigObject、HostConfigFile、および EnableWebAgent の各パラメータのほかに、以下のパラメータもフレームワーク エージェントの WebAgent.conf ファイルに追加されます。

重要: Web エージェント以外の他の SiteMinder 製品を参照しているファイルのセクションは変更しないでください。 ただし、ファイル内の Web エージェント パラメータの値は変更できます。

LocalConfigFile

LocalConfig.conf ファイルのロケーションを指定します。このファイルに、エージェント設定の大半が含まれます。

ServerPath

エージェントに対して Web サーバ（Apache 2.0 および Oracle iPlanet Web サーバ）のディレクトリを特定します。

LoadPlugin

どのプラグインがフレームワーク エージェントに対してロードされるか指定します。 プラグインはさまざまな種類のエージェント機能をサポートします。 以下のプラグインを使用できます。

HttpPlugin

Web エージェントが HTTP エージェントとして動作するかどうかを指定します。

デフォルト： Enabled

SAMLAffiliatePlugin

Web エージェントと SAML アフィリエイト エージェントの間の通信を許可します（Federation セキュリティ サービスを購入している場合）。

デフォルト： Disabled

Affiliate10Plugin

Web エージェントと 4.x アフィリエイト エージェントの間の通信を許可します。

デフォルト： 無効。

制限： SAML アフィリエイト エージェントはこのプラグインを使用しません。

OpenIDPlugin

Web エージェントが OpenID 認証方式（OIAS）を使用するようにします。

デフォルト： Disabled

他の LoadPlugin エントリを有効にするには、行の先頭からポンド記号（#）を削除します。

AgentIdFile

エージェントの一意の ID 文字列を格納する AgentId ファイルのパスを指定します。 エージェントは自動的に AgentId ファイルを生成しますが、これを変更してはいけません。 AgentId ファイルを更新するためには、エージェントは Windows でも UNIX でも書き込み許可が必要です。 Windows では、Web エージェント設定ウィザードは自動的に書き込み許可を与えます。

デフォルトの名前： Agentid.dat

パス： WebAgent.conf ディレクトリ/AgentId.dat

詳細情報

複数の Web サーバ インスタンスを持つ Web エージェントの管理 (P. 66)

LocalConfig.conf ファイルの場所（フレームワーク エージェント）

フレームワーク Web エージェントをインストールすると、SiteMinder インストール プログラムは、以下のディレクトリに LocalConfig.conf ファイルを作成します。

Windows

web _agent_home¥config

UNIX

web _agent_home/config

重要: このファイルにはすべてのデフォルト設定が含まれています。このファイルを直接変更しないでください。 後で参照したりリカバリしたりするために、このファイルのバックアップ コピーを作成することをお勧めします。

Web エージェントを設定するときに、設定ウィザードにより、LocalConfig.conf ファイルは以下のディレクトリにコピーされます。

IIS Web サーバ

web_agent_home¥bin¥IIS

Oracle iPlanet Web サーバ

Oracle_iPlanet_home/https-*hostname*/config

Apache Web サーバ

Apache_home/conf

Web エージェントは、LocalConfig.conf ファイルのこのコピーからその設定を取得します。

ローカル設定ファイルのみにあるパラメータ

中央エージェント設定の場合、ローカル設定ファイル内のパラメータの大半は、エージェント設定オブジェクトにも含まれています。 以下のパラメータは、ローカル設定ファイルのみで使用され、エージェント設定オブジェクトには*ありません*。

AgentConfigObject

ローカル エージェント設定ファイル内にエージェント設定オブジェクト (ポリシー サーバに格納された) の名前を定義します。このパラメータはエージェント設定オブジェクトでは*使用されません*。

デフォルト： デフォルトなし

EnableWebAgent

Web エージェントをアクティブにし、それがポリシー サーバと通信することを可能にします。 すべての設定パラメータの変更を完了してから、このパラメータを yes に設定します。

デフォルト： No

HostConfigFile

トラステッド ホスト コンピュータがポリシー サーバに正常に登録された後に作成される SMHost.conf ファイル（IIS 6.0 または Apache のエージェント内）のパスを指定します。 コンピュータ上のすべての Web エージェントが SMHost.conf ファイルを共有します。

デフォルト： デフォルトなし

ローカル設定の実装

以下のパラメータを使用して、ローカル設定が許可されるかどうかを制御できます。

AllowLocalConfig

ローカル設定ファイルを読み取ってエージェントの設定パラメータを取得するように、ポリシー サーバ上のエージェント設定オブジェクトに指示します。 このパラメータはエージェント設定オブジェクトでのみ使用されます。

ローカル設定ファイルで変更可能なパラメータを制御するために、このパラメータの複数の値をエージェント設定オブジェクトに追加します。 複数の値がこのパラメータに設定される場合、それらは以下の順序で処理されます。

- Yes が使用される場合、パラメータはすべてローカルで設定できます。
- パラメータのリストに対しては No が優先されます。 また、両方の値が一緒に設定された場合は、No が Yes を上書きします。このオプションにより、エージェント設定オブジェクトの他の設定パラメータのいずれも削除せずに、迅速かつ完全にローカル設定を無効にすることができます。

デフォルト： No （ローカル設定は禁止されています）

例： No, EnableAuditing, EnableMonitoring （すべてのローカル設定は禁止されています）

例： No, Yes （すべてのローカル設定は禁止されています）

例： EnableAuditing, EnableMonitoring （前の 2 つのパラメータについてのみ、ローカル制御が可能です）

ローカル設定を実装する方法

1. 管理 UI にログインします。

   ［Welcome］画面が表示されます。

2. ［インフラストラクチャ］-［エージェント設定オブジェクト］をクリックします。

   エージェント設定オブジェクトのリストが表示されます。

3. 目的のエージェント設定オブジェクトの行で、変更アイコンをクリックします。

［エージェント設定の変更］ダイアログ ボックスが表示されます。

4. AllowLocalConfig パラメータの左側の編集アイコンをクリックします。

   ［パラメータの編集］ダイアログ ボックスが表示されます。

5. ［値］フィールドのテキストを yes に変更し、［OK］をクリックします。

   ［パラメータの編集］ダイアログ ボックスが閉じます。

6. ［サブミット］をクリックします。

   確認メッセージが表示されます。

7. （オプション）将来の参照用に、変更に関する任意の注釈を［コメント］フィールドに入力します。

8. ［はい］をクリックします。

   ローカル設定が有効になります。

9. Web サーバ上で該当するローカル設定ファイルを開き、対象のパラメータ設定を変更します。

10. 従来のエージェントに限り、EnableWebAgent パラメータの値を yes に設定します。

11. ローカル設定ファイルを保存して閉じます。

12. Framework エージェントに限り、以下の手順に従います。

    a. WebAgent.conf ファイルを開きます。

    b. EnableWebAgent パラメータの値を yes に設定します。

    c. WebAgent.conf ファイルを保存して閉じます。

13. Web サーバを再起動します。

    ローカル設定が有効になり、更新されたすべてのパラメータが変更されます。

詳細情報:

Web エージェントの有効化 (P. 77)
変更時にサーバの再起動を必要とするパラメータ (P. 27)

エージェント設定ファイルを編集する方法

エージェント設定ファイルは、ローカルで設定された Web エージェントの設定を制御します。 それらの設定を変更するには、以下の手順に従います。

1. WebAgent.conf （従来のエージェントの場合）または LocalConfig.conf ファイル（フレームワーク エージェントの場合）のバックアップ コピーを作成します。
2. テキスト エディタで元のエージェント設定ファイルを開きます。
3. 以下のいずれかを実行することによって、パラメータを有効または無効にします。
   - パラメータを有効にするには、行の先頭からポンド記号（#）を削除する。
   - パラメータを無効にするには、行の先頭にポンド記号（#）を追加する。
4. 以下のガイドラインを使用して、パラメータの値を変更します。
   - パラメータ名、等号（=）、パラメータ値の間に空白を挿入しないでください。
   - パラメータ値を引用符で囲みます。
   - WebAgent.conf ファイルおよび LocalConfig.conf ファイルでは、大文字と小文字が区別されません。エージェントと共にインストールされるサンプル ファイルについては、大文字小文字を区別する必要はありません。
   - 多くの値は、ファイル内で <Agent Name>,<IP Address> のようにわかりやすい変数の形で記載されています。 山形かっこ <> とテキストの両方を、希望の値に置き換えます。
   - 値が空白の場合、空白はデフォルトとして有効です。パラメータの先頭にポンド記号 (#) が記述されていない場合に限って、デフォルト値が適用されます。
5. 作業が完了した場合にのみ、EnableWebAgent を yes に設定します。 その後、ファイルを保存して閉じます。

   すべてのローカル設定変更が有効になります。エージェントを有効にした後にさらに変更を加えた場合は、それらの変更を適用するために Web サーバを再起動します。

ローカル設定パラメータの変更の制限

中央設定のエージェントでは、ローカル Web サーバ管理者の変更する設定パラメータを、ユーザが制限できます。 SiteMinder 管理者と Web サーバ管理者が別の人物である場合は、このメソッドが推奨されます。

次の手順に従ってください:

1. 管理 UI にログインします。

   ［Welcome］画面が表示されます。

2. ［インフラストラクチャ］-［エージェント設定オブジェクト］をクリックします。

   エージェント設定オブジェクトのリストが表示されます。

   目的の［エージェント設定オブジェクト］の行で編集アイコンをクリックします。

   ［エージェント設定の変更］ダイアログ ボックスが表示されます。

3. AllowLocalConfig パラメータの左側の編集アイコンをクリックします。

   ［パラメータの編集］ダイアログ ボックスが表示されます。

4. ［値］フィールドのテキストを消去し、次に、複数値オプション ボタンをクリックします。

5. ［追加］をクリックします。

   空のフィールドが表示されます。

6. フィールドでのアクセスを許可するパラメータの名前を入力します。複数のパラメータはカンマで区切ります。 リスト内のそれらのパラメータのみ、ローカルで変更できます。

   例： 以下の例では、ローカル Web サーバ上で EnableAuditing および EnableMonitoring パラメータのみを設定できるようにする方法を示します。

   AllowLocalConfig=EnableAuditing,EnableMonitoring

7. （オプション）手順 5 ～ 6 を繰り返して、さらにパラメータを追加します。

8. ［OK］をクリックします。

   ［パラメータの編集］ダイアログ ボックスが閉じ、［エージェント設定の変更］ダイアログ ボックスが表示されます。

9. ［サブミット］をクリックします。

［エージェント設定の変更］ダイアログ ボックスが閉じ、確認メッセージが表示されます。

10. （オプション）将来の参照用に、変更に関する任意の注釈を［コメント］フィールドに入力します。
11. ［はい］をクリックします。

    変更は、次回 Web エージェントがポリシー サーバをポーリングしたときに適用されます。

中央設定とローカルの設定の組み合わせ

中央で設定したい Web エージェントの数が多いけれども、それらのうちの少数の Web エージェントの設定を他の Web エージェントの設定と変える必要がある場合は、中央とローカルの設定を組み合わせて使用することができます。

たとえば、エージェントを個別に設定せずに、SiteMinder ネットワークを介して複数の cookie ドメインのシングル サインオンを設定する必要がある場合は、すべてのエージェントに中央設定を使用して、別の設定を必要とする少数のグループにローカル設定を使用することができます。

前述の例で、エージェント設定オブジェクトの CookieDomain パラメータが example.com に設定されているとします。 ただし、ネットワーク内の 1 つの Web エージェントでは、CookieDomain パラメータを .example.net に設定し、その他のすべてのパラメータにはエージェント設定オブジェクトに設定されている値をそのまま使用する必要があります。

このサンプル設定を実装する方法

1. 管理 UI を使用して、環境に必要なパラメータをすべて指定してエージェント設定オブジェクトを作成します。 CookieDomain パラメータを .example.com に設定します。
2. エージェント設定オブジェクトの AllowLocalConfig パラメータを yes に設定します。
3. 1 つの Web エージェントで、CookieDomain パラメータの値として example.net を使用するように（Web サーバ上の）ローカル設定ファイルのみを変更します。 他のパラメータはいずれも変更 *しない*でください。

その単独のエージェントのローカル設定ファイル内の CookieDomain パラメータの値は、エージェント設定オブジェクト内の値より優先されますが、他のすべてのパラメータに関しては、エージェント設定オブジェクトによって設定値が決まります。

第 3 章: Web エージェントで使用される設定ファイル

SiteMinder Web エージェントでは、特定の設定について設定ファイルを使用します。これらの設定ファイルの一部は、Web エージェントと共に Web サーバにインストールされます。 他の設定ファイルは、SiteMinder Web エージェント設定ウィザードによって作成されます。これらの Web エージェント ファイルは、Web サーバをホストするコンピュータ上にインストールされた特定の Web サーバに関連付けられます。

たとえば、Apache Web サーバを実行する 32 ビット Windows システムに Web エージェントをインストールした場合、Web エージェント設定ウィザードは、SiteMinder Web エージェントによって必要とされる変更を既存の Apache Web サーバに加えます。

エージェント接続管理設定ファイル

Web エージェント インストール ウィザードは、エージェント接続マネージャ設定ファイル（AgentConMgr.conf）を以下の場所にインストールします。

web_agent_home/config

web_agent_home

SiteMinder エージェントがインストールされているディレクトリを示します。

デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）：C:¥Program Files¥CA¥webagent

デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）：C:¥Program Files¥CA¥webagent¥win64

デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）：C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）：/opt/ca/webagent

このエージェント接続マネージャ設定ファイルによって、Web エージェントが動作中に接続に関する詳細なトレース ログを作成することが可能になります。

詳細情報:

Agent Connection Manager のトレース ログによる詳細なエージェント接続データの収集 (P. 456)

Connection API 設定ファイル

Connection API ファイル（conapi.conf）は、Connection API によってサービスを設定するために使用されます。 これらのサービスには OneView モニタが含まれます。

Web エージェント インストール ウィザードは、Connection API 設定ファイルを以下の場所に作成します。

web_agent_home/config

web_agent_home

SiteMinder エージェントがインストールされているディレクトリを示します。

デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）： C:¥Program Files¥CA¥webagent

デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）： C:¥Program Files¥CA¥webagent¥win64

デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）： C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）： /opt/ca/webagent

注: OneView モニタの使用の詳細については、「DNA Always Current Scheduler」を参照してください。

ローカル エージェント設定ファイル

Web エージェント インストール ウィザードは、ローカル エージェント設定ファイル（LocalConfig.conf）を以下の場所にインストールします。

web_agent_home/config

web_agent_home

SiteMinder エージェントがインストールされているディレクトリを示します。

デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）：C:¥Program Files¥CA¥webagent

デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）：C:¥Program Files¥CA¥webagent¥win64

デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）：C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）：/opt/ca/webagent

このファイルによって、Web エージェントがインストールされているのと同じ Web サーバ上に Web エージェント設定パラメータを設定することができます。そのため、関連するポリシー サーバ上のエージェント設定オブジェクトに格納されているパラメータを使用する必要はありません。

IIS Web エージェントの場合、Web エージェント設定ウィザードでは、ローカル エージェント設定ファイルの重複コピーを以下の場所に作成します。

web_agent_home¥bin¥IIS

詳細情報:

ローカル エージェント設定 (P. 35)
LocalConfig.conf ファイルの場所（フレームワーク エージェント） (P. 40)

トレース設定ファイル

トレース設定ファイル(trace.conf)を使用して、以下の項目についてトレース ログを設定できます。

- 接続 API
- IPC プロバイダ
- TCP/IP （Transport）
- API の監視

Web エージェント インストール ウィザードは、トレース設定ファイルを以下の場所に作成します。

web_agent_home

SiteMinder エージェントがインストールされているディレクトリを示します。

デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）： C:¥Program Files¥CA¥webagent

デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）： C:¥Program Files¥CA¥webagent¥win64

デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）： C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）： /opt/ca/webagent

詳細情報:

IPC セマフォ関連メッセージ出力の Apache エラー ログへの制限 (P. 373)

Web エージェント トレース設定ファイル

Web エージェント トレース設定ファイルを使用して、Web エージェント操作のさまざまな要素についてトレース ログを作成することができます。たとえば、SiteMinder シングル サインオン（SSO）機能に関連するさまざまな Web エージェント動作についてトレース ログを作成できます。

Web エージェント インストール ウィザードは、Web エージェント トレース設定ファイルを以下の場所に作成します。

web_agent_home

SiteMinder エージェントがインストールされているディレクトリを示します。

デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）：C:¥Program Files¥CA¥webagent

デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）：C:¥Program Files¥CA¥webagent¥win64

デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）：C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）：/opt/ca/webagent

詳細情報:

トレース ロギングをセットアップする方法 (P. 441)

SiteMinder ホスト設定ファイル

Web エージェント設定ウィザードは、ホスト設定ファイル（SmHost.conf）を、SiteMinder Web エージェントが設定された Web サーバごとに、以下の場所に作成します。

web_agent_home/config

web_agent_home

SiteMinder エージェントがインストールされているディレクトリを示します。

デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）：C:¥Program Files¥CA¥webagent

デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）：C:¥Program Files¥CA¥webagent¥win64

デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）：C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）：/opt/ca/webagent

SmHost.conf ファイルには、Web エージェントが関連付けられているポリシー サーバに対する最初の接続で使用される情報が含まれます。

注: 詳細については、「SiteMinder Web エージェント インストール ガイド」を参照してください。

Web エージェント設定ファイル

SiteMinder Web エージェント設定ウィザードは、Web エージェント設定ファイル（WebAgent.conf）を、SiteMinder Web エージェントが設定された Web サーバごとに、以下の場所に作成します。

web_agent_home¥conf

web_agent_home

SiteMinder エージェントがインストールされているディレクトリを示します。

デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）：C:¥Program Files¥CA¥webagent

デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）：C:¥Program Files¥CA¥webagent¥win64

デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）：C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）：/opt/ca/webagent

このファイルを使用して、Web エージェントを有効または無効（開始または停止）することができます。

IIS Web エージェントの場合、Web エージェント設定ウィザードでは、ローカル エージェント設定ファイルの重複コピーを以下の場所に作成します。

web_agent_home¥bin¥IIS

詳細情報:

Web エージェントの有効化 (P. 77)
Web エージェントの無効化 (P. 78)

第 4 章: 基本的なエージェント セットアップおよびポリシー サーバ接続

このセクションには、以下のトピックが含まれています。

Web エージェント設定パラメータのデフォルト設定

別の値が指定されている場合を除き、Web エージェント設定パラメータのデフォルト設定が常に使用されます。

エージェント設定オブジェクトにもローカル設定ファイルにもパラメータがない場合は、デフォルト値が使用されます。

AgentName と DefaultAgentName の値の設定

AgentName パラメータは、エージェントのアイデンティティを指定します。 ポリシー サーバは、この識別情報を使用してポリシーを Web エージェントに関連付けます。 以下のパラメータを使用してエージェントの名前を定義できます。

AgentName

Web エージェントの ID を定義します。 この ID は、エージェントをホストしている各 Web サーバ インスタンスの名前と IP アドレスまたは FQDN をリンクします。

以下のイベントのいずれか発生した場合は、DefaultAgentName の値が AgentName パラメータの代わりに使用されます。

- AgentName パラメータが無効。
- AgentName パラメータの値が空。
- AgentName パラメータの値が既存のエージェント オブジェクトに一致しない。

注: このパラメータは複数の値を持つことができます。 エージェント設定オブジェクトでこのパラメータを設定する場合は、複数値オプションを使用します。 ローカル設定ファイルについては、ファイル内の個別の行に各値を追加します。

デフォルト： デフォルトなし

制限： 複数の値が許可されていますが、各 AgentName パラメータは 4,000 文字に制限されています。 文字をパラメータ名に追加することにより、必要に応じて追加の AgentName パラメータを作成します。 たとえば、AgentName、AgentName1、AgentName2 などを作成します。

制限： 32-127 の範囲内に 7 ビット ASCII 文字が含まれている必要があり、1 つ以上の印刷可能文字が含まれている必要があります。アンパサンド（&）およびアスタリスク（*）文字は含めることができません。 この値は大文字と小文字が区別されます。 たとえば、MyAgent と myagent という名前は、同じように処理されます。

例： myagent1,192.168.0.0 (IPV4)

例： myagent2, 2001:DB8::/32 (IPV6)

例： myagent,www.example.com

例（複数の AgentName パラメータ）： AgentName1、AgentName2、AgentName3。 各 AgentName*number* パラメータの値は、4,000 文字に制限されています。

DefaultAgentName

要求を処理するためにエージェントが使用する名前を定義します。エージェント名値が AgentName パラメータに存在しないときは、DefaultAgentName の値が IP アドレスまたはインターフェース上の要求に使用されます。

仮想サーバを使用している場合は、DefaultAgentName を使用することにより SiteMinder 環境を迅速にセットアップできます。DefaultAgentName を使用することは、各仮想サーバに対して個別のエージェントを定義する必要がないことを意味します。

重要: DefaultAgentName パラメータの値を指定しない場合、AgentName パラメータの値にはそのリスト内のすべてのエージェント ID が必要です。そうでない場合、ポリシー サーバはエージェントにポリシーを結び付けることができません。

デフォルト： デフォルトなし

制限： 複数の値を指定できます。

制限： 32-127 の範囲内に 7 ビット ASCII 文字が含まれている必要があり、1 つ以上の印刷可能文字が含まれている必要があります。アンパサンド（&）およびアスタリスク（*）文字は含めることができません。 この値は大文字と小文字が区別されます。 たとえば、MyAgent と myagent という名前は、同じように処理されます。

仮想サーバ サポートを設定する場合は、AgentName パラメータまたは DefaultAgentName パラメータのいずれかに対して、値を指定します。

次の手順に従ってください:

1. 次のいずれかの手順に従って AgentName の値を指定します。
   - 集中的なエージェント設定の場合は、管理 UI 上でエージェント設定オブジェクトを開き、目的の値を AgentName パラメータに追加します。
   - ローカル エージェント設定の場合は　Web サーバ上でローカル設定ファイルを開きます。 目的の値をファイル内の個別の行に追加します。
2. 次のいずれかの手順に従って、DefaultAgentName のアイデンティティを指定します。
   - 中央エージェント設定の場合は、管理 UI 上でエージェント設定オブジェクトを開き、目的の値を DefaultAgentName パラメータに追加します。

- ローカル エージェント設定の場合は　Web サーバ上でローカル設定ファイルを開きます。 目的の値を DefaultAgentName パラメータに追加します。

AgentName と DefaultAgentName の値が設定されます。

詳細情報

仮想サーバ サポートをセット アップする方法 (P. 186)

ローカル設定パラメータの変更の制限

中央設定のエージェントでは、ローカル Web サーバ管理者の変更する設定パラメータを、ユーザが制限できます。 SiteMinder 管理者と Web サーバ管理者が別の人物である場合は、このメソッドが推奨されます。

次の手順に従ってください:

1. 管理 UI にログインします。

   ［Welcome］画面が表示されます。

2. ［インフラストラクチャ］-［エージェント設定オブジェクト］をクリックします。

   エージェント設定オブジェクトのリストが表示されます。

   目的の［エージェント設定オブジェクト］の行で編集アイコンをクリックします。

   ［エージェント設定の変更］ダイアログ ボックスが表示されます。

3. AllowLocalConfig パラメータの左側の編集アイコンをクリックします。

   ［パラメータの編集］ダイアログ ボックスが表示されます。

4. ［値］フィールドのテキストを消去し、次に、複数値オプション ボタンをクリックします。

5. ［追加］をクリックします。

   空のフィールドが表示されます。

6. フィールドでのアクセスを許可するパラメータの名前を入力します。 複数のパラメータはカンマで区切ります。 リスト内のそれらのパラメータのみ、ローカルで変更できます。

   例： 以下の例では、ローカル Web サーバ上で EnableAuditing および EnableMonitoring パラメータのみを設定できるようにする方法を示します。

   AllowLocalConfig=EnableAuditing,EnableMonitoring

7. （オプション）手順 5 ～ 6 を繰り返して、さらにパラメータを追加します。

8. ［OK］をクリックします。

   ［パラメータの編集］ダイアログ ボックスが閉じ、［エージェント設定の変更］ダイアログ ボックスが表示されます。

9. ［サブミット］をクリックします。

［エージェント設定の変更］ダイアログ ボックスが閉じ、確認メッセージが表示されます。

10. （オプション）将来の参照用に、変更に関する任意の注釈を［コメント］フィールドに入力します。
11. ［はい］をクリックします。

    変更は、次回 Web エージェントがポリシー サーバをポーリングしたときに適用されます。

エージェント名の一致の確認

SiteMinder のルールおよびポリシーは、エージェント名に結び付けられています。 あるホストに対して要求を送信し、そのホストが持つエージェント名がポリシー サーバ上で不明の場合、ポリシー サーバはポリシーを実装できません。 したがって、Web エージェントの DefaultAgentName パラメータまたは AgentName パラメータの値は、ポリシー サーバ上で定義されたエージェント エントリの名前と一致している必要があります。

エージェントは、管理 UI を使用してポリシー サーバで定義します。［エージェント プロパティ］ダイアログ ボックスの［名前］フィールドに入力する値は、DefaultAgentName または AgentName 設定項目で定義された値と一致している必要があります。それぞれは、Web エージェントがローカルで（エージェント設定ファイル）、またはポリシー サーバから集中的に（エージェント設定オブジェクト）設定されていることを意味します。

エージェント名の暗号化

URL により、ユーザがフォーム、SSL、または NTLM 認証情報コレクタへリダイレクトされる場合、Web エージェントはデフォルトで、その URL に自らの名前を追加します。 EncryptAgentName パラメータを使用して、エージェントが URL 内のその名前を暗号化するかどうかや、認証情報コレクタが URL を受信する場合に名前を復号化するかどうかを制御できます。

EncryptAgentName パラメータのデフォルト設定は yes です。以下のいずれかの状況では、このパラメータを no に設定してください。

- 認証情報コレクタと共にサードパーティのアプリケーションを使用していて、そのアプリケーションが処理を進めるためにエージェント名を読み取れるようにする必要がある場合
- フォーム認証を実行し、認証の対象となる 1 つのリソースへユーザを振り向けるために、Web エージェントをフォーム認証情報コレクタ（FCC）として構成する場合 シングル リソース ターゲットを設定する手順では、暗号化されていないエージェント名が必要です。

Web エージェント名を暗号化するには、EncryptAgentName パラメータを yes に設定します。

詳細情報

シングル リソース ターゲットを使用するための FCC の設定 (P. 200)

Web エージェントとポリシー サーバ間の通信を管理する方法

以下のいずれかの手順を使用して、エージェントとポリシー サーバの間の通信を管理できます。

- ネットワーク遅延問題へ対応します (P. 65)。
- 複数の Web サーバ インスタンスでエージェントを管理します (P. 66)。

詳細情報:

Web エージェントの監視 (P. 420)

ネットワーク遅延への対応

ネットワーク遅延の問題が存在する場合、Web エージェントはポリシー サーバに接続できません。 この問題を回避するには、エージェント設定オブジェクトまたはローカル設定ファイル内で以下のパラメータを使用します。

AgentWaitTime

Low Level Agent Worker Process （LLAWP）が使用可能になるまで、エージェントが何秒待つかを指定します。 指定した時間を過ぎると、エージェントはポリシー サーバに接続しようとします。

このパラメータの設定は、LLAWP 接続に関連するエージェント スタートアップ エラーを解決するのに役立つ場合があります。 デフォルト値で設定を開始し、エージェントが正常に起動されるまで、5 秒ずつ間隔を増やすことをお勧めします。
ローカル設定を使用している場合は、エージェント設定オブジェクト *ではなく* WebAgent.conf ファイルでこのパラメータを設定します。

デフォルト： 5

例： 以下の式を使用して推奨値を計算します。

（*The_number_of_Policy_Servers* x 30） + 10 = AgentWaitTime パラメータの値（秒単位）。

たとえば、5 つのポリシー サーバがある場合は、AgentWaitTime パラメータの値を 160 に設定します。［（5x30）+ 10 = 160］（秒）。

制限： （FIPS 互換モードおよび FIPS 移行モード） 5 以上。

制限： （FIPS 専用モード） 20 以上。

設定値を大きくするのは、ネットワーク遅延の問題が存在する場合のみにしてください。 設定値を大きくすると、予期しない Web サーバの動作が発生する可能性があります。

ネットワーク遅延に対応するには、エージェント設定オブジェクトまたはローカル設定ファイルで AgentWaitTime パラメータを有効にします。次に、目的の秒数を指定します。

複数の Web サーバ インスタンスを持つ Web エージェントの管理

複数の Web サーバ インスタンス上に Web エージェントを設定する場合、各サーバ インスタンスは、独自の Web エージェント キャッシュ、ログ ファイル、および状態監視リソースを持つ必要があります。 リソースが一意であることを確認するには、WebAgent.Conf ファイルに以下のパラメータを設定します。

ServerPath

Web エージェントが Web サーバの複数インスタンスを使用するように設定されている場合に、各 Web サーバ インスタンスの一意のパスを指定します。 ServerPath は、エージェントが使用するリソースのキャッシュ、ロギング、および状態監視について、一意の識別子を作成します。

この値は、システム上で実行されているサーバ インスタンス間で一意の英数文字列である必要があります。 たとえば、2 つのサーバ インスタンスがある場合、一方のインスタンスの ServerPath パラメータの値を MyAgent1 に設定し、もう一方のインスタンスの値を MyAgent2 に設定できます。

デフォルト： 空白

例： 4 つの Web サーバ インスタンスがあり、それぞれがエージェントをロードする場合、各サーバ インスタンスの WebAgent.conf ファイルの ServerPath パラメータには一意の値を設定します。 ServerPath パラメータは *server_instance_root*/logs など、各サーバ インスタンスのログ ファイルが保管されるディレクトリに設定できます。

複数のサーバ インスタンス上に Web エージェントを設定するには、各 WebAgent.conf ファイルの ServerPath パラメータに一意のパスを追加します。

Windows システムに関する ServerPath パラメータの設定

ユーザの Windows オペレーティング環境に複数のサーバ インスタンスがある場合は、WebAgent.conf ファイルで以下のパラメータに対する値を指定します。

ServerPath

Web エージェントが Web サーバの複数インスタンスを使用するように設定されている場合に、各 Web サーバ インスタンスの一意のパスを指定します。 ServerPath は、エージェントが使用するリソースのキャッシュ、ロギング、および状態監視について、一意の識別子を作成します。

この値は、システム上で実行されているサーバ インスタンス間で一意の英数文字列である必要があります。 たとえば、2 つのサーバ インスタンスがある場合、一方のインスタンスの ServerPath パラメータの値を MyAgent1 に設定し、もう一方のインスタンスの値を MyAgent2 に設定できます。

デフォルト： 空白

例： 4 つの Web サーバ インスタンスがあり、それぞれがエージェントをロードする場合、各サーバ インスタンスの WebAgent.conf ファイルの ServerPath パラメータには一意の値を設定します。 ServerPath パラメータは *server_instance_root*/logs など、各サーバ インスタンスのログ ファイルが保管されるディレクトリに設定できます。

注: ServerPath パラメータに指定する文字列に円記号（¥）を使用しないでください。その他の文字はすべて使用できます。

ServerPath パラメータは以下の Windows プラットフォームには必要*ありません*。

- IIS サーバ（常にサーバ インスタンスが 1 つのみあります）。
- Apache 2.0 サーバ （Web サーバ インスタンスが 1 つのみある場合）。このパラメータはこれらのシステム上でサポートされていますが、使用されてはいません。
- Oracle iPlanet または Domino Web サーバ

  これらのサーバは Windows 上でマルチプロセス モードで実行*されません*。

UNIX システムに関する ServerPath パラメータの設定

ServerPath パラメータは WebAgent.conf ファイル内にあります。

UNIX プラットフォーム上の Web サーバでは、各サーバ インスタンスが独自のエージェント リソースを持つようにすることをお勧めします。

UNIX 上の以下のサーバについて、ServerPath パラメータを設定します。

- Apache 2.0 （IBM HTTP Server など、すべての Apache 2.0 ベースのサーバを含む）
- Oracle iPlanet Web サーバ インスタンス

注: ServerPath は、UNIX システム上の Domino Web サーバには必要*ありません*。

ServerPath パラメータに設定する値は、システム上で実行されているサーバ インスタンス間で一意の英数字文字列である必要があります。 たとえば、2 つのサーバ インスタンスがある場合、一方のインスタンスの ServerPath パラメータの値を MyAgent1 に設定し、もう一方のインスタンスの値を MyAgent2 に設定できます。

ServerPath パラメータを必要とする追加設定

次のリストでは、ServerPath パラメータを必要とする他のユース ケースについて説明します。

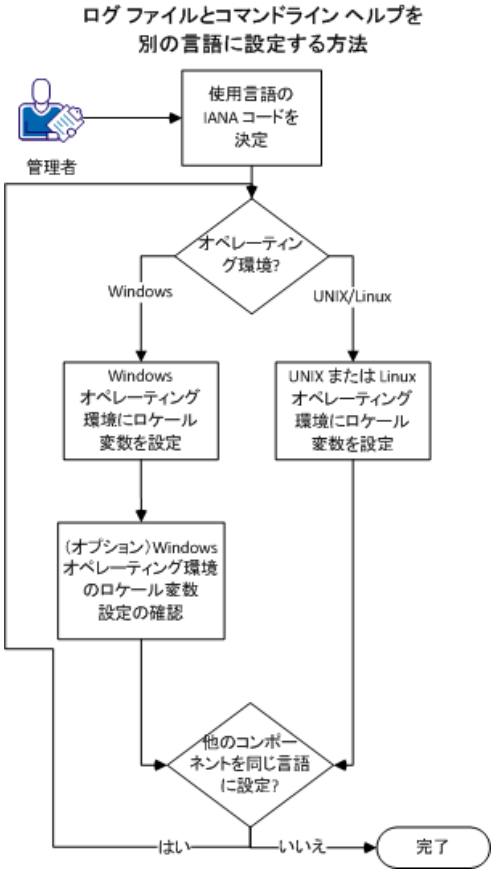
- Web エージェントは、1 つのセマフォを使用して共有メモリを制御しています。 セマフォはオペレーティング システム（またはカーネル）ストレージ内の値です。 システム上で実行されるプロセスは、リソースの可用性を確認するためにこの値を調べます。 セマフォが一意ではないので、複数のエージェントがメモリの同じ領域を指そうとします。 サーバ パスを指定した場合はインスタンスのルートが決定され、エージェントはセマフォに関連して固有のキーを作成するためのファイルを見つけることができます。
- サーバ インスタンス（Windows を除くすべてのプラットフォーム）が複数あると、エージェントは以下のいずれかの実行に失敗します。
  - AgentName パラメータの値の暗号化（00-0012 のエラー）。
  - SMSESSION Cookie または SMIDENTITY Cookie の暗号化。
  - エージェントが開始するときに、エージェント暗号化キーを更新すること。
- Apache では（Windows を除くすべてのプラットフォーム）、Apache が再起動したときに、エージェントは 6 つの共有メモリ セグメント（セマフォ）を解放しません。
- Web エージェントがそれぞれ Apache 2.0 サーバおよび Oracle iPlanet サーバなど、同じシステム上の別の Web サーバ タイプに対して設定される場合。 各サーバの設定に対する一意の ServerPath 値を指定します。 異なる Web サーバ タイプ同士では、エージェント リソースを共有 *できません*。

ログ ファイルの設定方法、および別の言語へのコマンドライン ヘルプ

以下のコンポーネントは、ログ ファイル、およびその他の言語でのコマンドライン ヘルプをサポートします。

- ポリシー サーバ
- Web エージェント
- レポート サーバ
- CA SiteMinder Agent for SharePoint
- CA SiteMinder for Secure Proxy Server
- <エージェント>
- SiteMinder SDK で作成される任意のカスタム ソフトウェア。

以下の図は、ログ ファイル設定のワークフローおよび，別の言語へのコマンドライン ヘルプについて説明しています。



次の手順に従ってください:

1. 言語の IANA コードを決定します (P. 72)。
2. 以下のいずれかの手順を使用して、オペレーティング環境用の環境変数を作成します。
   - Windows オペレーティング環境上でロケール変数を設定します (P. 74)。
   - UNIX または Linux オペレーティング環境で、ロケール変数を設定します (P. 76)。
3. （オプション）ウィンドウ オペレーティング環境で、ロケール変数の設定を確認します (P. 75)。
4. （オプション）手順 1 ～ 3 を繰り返して、ユーザの環境内の他のコンポーネントを同じ言語に設定します。

ユーザの言語の IANA コードを決定します。

各言語にはそれぞれ一意のコードがあります。 IANA（インターネット番号割当機関）は、これらの言語コードを割り当てます。 言語コードをロケール変数に追加することで、ソフトウェアが表示する言語を変更します。ロケール変数を作成する前に、目的の言語に該当するコードを決定します。

以下の表は、このソフトウェアでサポートされている言語に対応する IANA コードのリストを示しています。

言語	IANA コード
ポルトガル語（ブラジル）	pt_BR
フランス語	fr
ドイツ語	de
イタリア語	it
日本語	ja
韓国語	ko
中国語（簡体字）	zh-Hans
スペイン語	es

注: IANA 言語コードのリストは、このサードパーティ Web サイトから利用可能です。

環境変数

環境変数は、ユーザのニーズに適合するように、ユーザがコンピュータをカスタマイズできる設定です。 この環境変数の例には、以下のような項目があります。

- ダウンロードされたファイルを検索または格納するためのデフォルトディレクトリ。
- ユーザ名。
- 実行可能ファイルを検索する場所のリスト（パス）。

Windows オペレーティング環境ではグローバル環境変数を設定でき、これはコンピュータのすべてのユーザに適用されます。UNIX または Linux オペレーティング環境での環境変数は、各ユーザまたをプログラムに対して設定する必要があります。

ロケール変数を設定するには、以下のリストからユーザのオペレーティング環境用の手順を選択します。

- Windows オペレーティング環境上でロケール変数を設定します (P. 74)。
- UNIX または Linux オペレーティング環境で、ロケール変数を設定します (P. 76)。

Windows オペレーティング環境でのロケール変数の設定

以下のロケール変数は、ソフトウェアの言語設定を指定します。

SM_ADMIN_LOCALE

この変数を作成し、それを目的の言語に設定します。 別の言語を使用する各コンポーネントで、この変数を設定します。たとえば、ポリシー サーバ、およびフランス語に設定されているエージェントがあると仮定します。それらのコンポーネントの両方で、この変数をフランス語に設定します。

注: インストールまたは設定プログラムでは、この変数は設定 *されません*。

次の手順に従ってください:

1. ［スタート］-［コントロール パネル］-［システム］-［システムの詳細設定］をクリックします。

   ［システムのプロパティ］ダイアログ ボックスが表示されます。
2. ［詳細設定］タブをクリックします。
3. ［環境変数］をクリックします。
4. ［システム変数］セクションを見つけてから、［新規］をクリックします。

   ［新しいシステム変数］ダイアログ ボックスが、カーソルが［変数名］フィールドにある状態で表示されます。
5. 以下のテキストを入力します。

   SM_ADMIN_LOCALE
6. ［変数名］フィールドをクリックしてから、目的の IANA 言語コード (P. 72) を入力します。
7. ［OK］をクリックします。

   ［新しいシステム変数］ダイアログ ボックスが閉じ、SM_ADMIN_LOCALE 変数がリストに表示されます。
8. ［OK］を *2 回*クリックします。

   ロケール変数が設定されます。
9. （オプション）手順 1 ～ 8 を繰り返して、他のコンポーネントを同じ言語に設定します。

Windows オペレーティング環境でのロケール変数値の確認

ロケール変数が設定される値は、随時変更できます。 この手順は、変数を設定して、それが適切に設定されていることを確認した後に実行できます。

注: UNIX および Linux での変数の検証手順は、「プロシージャの設定 (P. 76)」にあります。

次の手順に従ってください:

1. 以下の手順で、コマンドライン ウィンドウを開きます。
   a. ［スタート］-［ファイル名を指定して実行］をクリックします。
   b. 以下のコマンドを入力します。

      ```
      cmd
      ```

   c. ［OK］をクリックします。

      コマンドライン ウィンドウが開きます。
2. 以下のコマンドを入力します。

   ```
   echo %SM_ADMIN_LOCALE%
   ```

   ロケールが次の行に表示されます。 たとえば、言語がドイツ語に設定される場合、以下のコードが表示されます。

   ```
   de
   ```

   ロケール変数の値が確認されます。

UNIX または Linux オペレーティング環境でのロケール変数の設定

以下のロケール変数は、ソフトウェアの言語設定を指定します。

SM_ADMIN_LOCALE

この変数を作成し、それを目的の言語に設定します。 別の言語を使用する各コンポーネントで、この変数を設定します。たとえば、ポリシー サーバ、およびフランス語に設定されているエージェントがあると仮定します。それらのコンポーネントの両方で、この変数をフランス語に設定します。

注: インストールまたは設定プログラムでは、この変数は設定 *されません*。

次の手順に従ってください:

1. 目的のコンポーネントを実行しているコンピュータにログインします。
2. コンソール（コマンドライン）ウィンドウを開きます。
3. 以下のコマンドを入力します。

   export SM_ADMIN_LOCALE=*IANA_language_code*

   以下の例のコマンドは、言語をフランス語に設定します。

   export SM_ADMIN_LOCALE=fr

   ロケール変数が設定されます。
4. （オプション）以下のコマンドを入力して、ロケール変数が適切に設定されていることを確認します。

   echo $SM_ADMIN_LOCALE

   ロケールが次の行に表示されます。 たとえば、言語がドイツ語に設定される場合、以下のコードが表示されます。

   de
5. （オプション）手順 1 ～ 4 を繰り返して、他のコンポーネントを同じ言語に設定します。

第 5 章: Web エージェントの起動と停止

このセクションには、以下のトピックが含まれています。

詳細情報:

WebAgent.conf ファイルのロケーション (P. 36)

Web エージェントの有効化

エージェントのパラメータを設定して、エージェントが Web サーバ上のリソースを保護できるようにします。

注: SiteMinder ポリシー サーバにポリシーも定義するまでは、リソースは保護*されません*。

次の手順に従ってください:

1. WebAgent.conf ファイルをテキスト エディタで開きます。

   注: 64 ビット Windows オペレーティング環境にインストールされた IIS 用 SiteMinder 12.52 エージェントには *2* つの WebAgent.conf ファイルがあります。 1 つのファイルは 32 ビット Windows アプリケーションと関連付けられます。 もう 1 つのファイルは 64 ビット Windows アプリケーションと関連付けられます。 特定の IIS Web サーバ上のすべての 32 ビット*および*64 ビット アプリケーション上で IIS 用 SiteMinder エージェントを開始または停止するには、*両方の* WebAgent.conf ファイルを変更します。

2. EnableWebAgent パラメータの値を yes に変更します。
3. WebAgent.conf ファイルを保存して閉じます。
4. Web サーバ（サーバが実行されるコンピュータではなく Web サーバ自体）を再起動します。

   Web エージェントが有効になります。

Web エージェントの無効化

Web エージェントによる Web サーバ上のリソースの保護およびポリシーサーバとの通信を停止するには、Web エージェントを無効にします。

次の手順に従ってください:

1. WebAgent.conf ファイルをテキスト エディタで開きます。

   注: 64 ビット Windows オペレーティング環境にインストールされた IIS 用 SiteMinder 12.52 エージェントには *2* つの WebAgent.conf ファイルがあります。 1 つのファイルは 32 ビット Windows アプリケーションと関連付けられます。 もう 1 つのファイルは 64 ビット Windows アプリケーションと関連付けられます。 特定の IIS Web サーバ上のすべての 32 ビット*および*64 ビット アプリケーション上で IIS 用 SiteMinder エージェントを開始または停止するには、*両方の* WebAgent.conf ファイルを変更します。

2. EnableWebAgent パラメータの値を no に変更します。
3. WebAgent.conf ファイルを保存して閉じます。
4. Web サーバ（サーバが実行されるコンピュータではなく Web サーバ自体）を再起動します。

   Web エージェントが無効になります。

詳細情報:

WebAgent.conf ファイルのロケーション (P. 36)

apachectl コマンドによるほとんどの Apache ベース エージェントの起動または停止

UNIX または Linux オペレーティング環境で apachectl コマンドを使用してほとんどの Apache ベース エージェントを起動または停止するには、まず製品の環境変数を設定する必要があります。

注: Apache ベース エージェントは apachectl -restart オプションをサポートしていません。 この手順は Apche ベースの IBM HTTP Server には適用されません。 代わりに、この手順を実行します。

次の手順に従ってください:

1. UNIX/Linux オペレーティング環境では、以下のスクリプトを実行することにより環境変数を設定します。

   ```
   ./ca_wa_env.sh
   ```

2. 以下のいずれかのコマンドを使用します。

   ```
   apachectl -stop
   ```

   ```
   apachectl -start
   ```

第 6 章: ユーザ保護

このセクションには、以下のトピックが含まれています。

エージェントがポリシーまたはキーの更新をチェックする頻度の変更

Web エージェントは、以下のアイテムをチェックするために定期的にポリシー サーバをポーリングします。

- 更新された管理情報
- 更新されたポリシー
- 動的に更新されたエージェント キー

この間隔は必要に応じて以下のパラメータを使用して変更できます。

PSPollInterval

ポリシー変更に関する情報または動的に更新されたキーを取得するために Web エージェントがポリシー サーバと通信する間隔（秒単位）を指定します。 数値が大きい（間隔が長い）ほど、ネットワーク トラフィックは減少します。 数値が小さい（間隔が短い）ほど、ネットワーク トラフィックは増加します。

デフォルト： 30

制限： 1

Web エージェントが更新がないかポリシー サーバをチェックする頻度を変更するには、PSPollInterval パラメータの秒数を変更します。

重要: PSPollInterval パラメータを増加させると、Web エージェントで SiteMinder ポリシー変更が提供されるタイミングにも影響があります。たとえば、勤務が終了した従業員のアクセスを無効にするため、10:30 にポリシーを変更し、PSPollInterval パラメータの値が 3600（1 時間の秒数）であるとします。 この場合、Web エージェントでは、変更されたポリシーを 11:30 まで適用しません。

詳細情報:

Web エージェントとダイナミック キーのロールオーバー (P. 24)

ユーザ トラッキングおよび URL モニタリング

SiteMinder エージェントは以下の手順で解説されているパラメータを使用してユーザを追跡し、URL を監視できます。

- 匿名 レルム間でユーザ ID を追跡します (P. 83)。
- ユーザ アクティビティまたはアプリケーション使用状況のを追跡します (P. 84)。
- URL 監視の概要

匿名レルム間でのユーザ ID の追跡

匿名ユーザがリソースにアクセスする場合、そのユーザは、SMIDENTITY（匿名） cookie の割り当てを受けます。 ユーザは、他のドメインに移動するときに、認証情報を要求されます。正常にログインすると、SMSESSION（ログイン済み）cookie が割り当てられます。

このユーザは、保護された「匿名の」リソース、つまりユーザーが認証情報を提示する必要のないレルム内に存在しているリソースにアクセスする場合、1 人のユーザに対応する両方の cookie を保持している 1 つのドメインに入ることができます。5.x QMR 3 以降の Web エージェントによって保護されているリソースの場合、Web エージェントは SMIDENTITY cookie ではなく、SMSESSION cookie を使用してユーザを識別します。

ユーザが、完全にアップグレードされたドメインから、古いバージョンのエージェントによって SMIDENTITY cookie でユーザが識別されるドメインへと移動する場合、使用される cookie は要求を処理する Web エージェントのバージョンによって異なります。

別個の cookie ドメインに関しては、保護されているリソースがマスタ cookie ドメインに含まれ、匿名リソースが第 2 ドメインに含まれている場合、ユーザは、以下のタスクを行うと、引き続き匿名ドメインの匿名ユーザと見なされます。

1. 最初に匿名ドメインにアクセスします
2. マスタ ドメインに移動し、ログインします。
3. 匿名ドメインへ戻ります。

監査によるユーザ アクティビティまたはアプリケーション使用状況の追跡

監査によって、Web サイトでのアプリケーションの使用頻度を測定したり、ユーザ アクティビティを追跡したりすることができます。 監査は以下のパラメータを使用して制御されます。

EnableAuditing

ユーザ セッション キャッシュに格納される正常に行われたすべてのユーザ許可に関する情報を Web エージェントが記録するかどうかを指定します。 ユーザ許可を有効にすると、Web エージェントがポリシー サーバへ問い合わせを行わずにキャッシュの情報を使用しても、ユーザ許可情報は記録されます。 ユーザがリソースにアクセスすると、Web エージェントはユーザ名とアクセス情報を固有の Web サーバ ログファイルに記録します。

デフォルト： No

ポリシー サーバと Web エージェントの両方でユーザ アクティビティを監査します。 キャッシュに格納されている情報に基づいてユーザにリソースへのアクセスが許可されるたびに、Web エージェントからポリシー サーバ監査サービスにメッセージが送信されます。 このアクションにより、監査サービスは、ポリシー サーバと Web エージェントで正常に行われたユーザ許可を追跡します。 Web エージェントが監査メッセージを監査サービスに送信できなかった場合、リソースへのアクセスは拒否されます。 その後、管理 UI から SiteMinder アクティビティ レポートを実行できます。 ポリシー サーバからのレポートには、各 SiteMinder セッションのユーザ アクティビティが示されます。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

監査によってユーザ アクティビティまたはアプリケーション使用状況を追跡するには、EnableAuditing パラメータの値を yes に設定します。

URL 監視の概要

Web エージェントには、Web サイトの正常な運用を中断させようとしたり、サイトのセキュリティ機構を回避して情報に不正にアクセスしようとしたりするユーザからの攻撃に対する防備機能が備わっています。

Web エージェントでは、リソースリクエストの URL を監視して、対象となるリソースのセキュリティポリシーを実行します。 SiteMinder Web エージェントが URL を解釈および解析する方法は、リソースが置かれている Web サーバの方法と異なります。 このような違いがあるため、パフォーマンスが微妙に異なり、セキュリティの問題が発生する可能性があります。また、場合によっては無許可のユーザがリソースにアクセスできることもあります。 Web サイトの設計および SiteMinder Web エージェントの設定では、これらの問題を考慮する必要があります。

攻撃の防止

SiteMinder エージェントは以下の手順で解説されているパラメータを使用して、攻撃に対する防御を支援できます。

- クロスサイト スクリプティングから Web サイトを保護します (P. 87)。
- クロス サイト スクリプティングを防ぐためにエージェントを設定します (P. 89)。
- クロス サイト スクリプティングから J2EE アプリケーションを保護します (P. 90)。
- 有効なターゲット ドメインを定義します (P. 89)
- クロスサイト スクリプティング攻撃に対して J2EE アプリケーションを保護します (P. 90)
- CSS のデフォルト文字セットを上書きします (P. 91)。
- URL のクエリ文字列部分で特定文字を禁止します (P. 93)。
- URL で特定文字を禁止します (P. 95)。
- フォームで特定文字を禁止します (P. 97)。
- DNS サービス拒否攻撃を阻止します (P. 98)。
- P 拡張子が付いていないリソースまたはファイルを保護します (P. 99)。
- POST 維持を無効にします (P. 100)。
- アプリケーションをセキュリティ保護します (P. 101)。
- カスタム レスポンスが X-Frame Options に準拠していることを確認します (P. 102)

クロスサイト スクリプティングからの Web サイトの保護

クロスサイト スクリプティング（CSS）攻撃が発生するのは、ブラウザからの入力テキスト（通常は、ポストされたデータ、または URL 内のクエリ パラメータから得られたデータ）がブラウザによって表示される場合です。このとき、有効で実行可能なスクリプトを形成することが可能な文字を、フィルタ処理なしでブラウザ内で表示してしまうことが問題です。

疑いを抱いていないユーザに対して、攻撃用 URL を提示できます。 ユーザがその URL をクリックした場合、アプリケーションはブラウザに対して表示を返すことがあります。その表示の中には、入力文字と共に、クエリ文字列の中に含まれている無効な文字に関するエラー メッセージがあります。 ブラウザ上でこれらのパラメータに関する表示が実施された場合、望まれていなかったスクリプトがそのブラウザ上で実行される事態が発生する可能性があります。

たとえば、検索エンジンの Web ページでユーザが「news」と入力した場合、アプリケーションは通常、空白のフィールド、または以下のような応答を返すことがあります。

news の検索結果は以下のとおりです:

攻撃用 URL への応答として、ブラウザは次のような応答を受け取ることがあります。

news<script>BadProgram</script>

二重引用符が（"）が ASCII 文字として入力された場合、BadCSSChars パラメータはそれを解釈しません。 二重引用符を無効なクロスサイト スクリプティング文字として含める場合は、ASCII 文字と同等の 16 進数値である %22 を入力します。 例：

BadCSSChars="%22"

Web エージェント FCC ページのクロスサイト スクリプティング攻撃の防止

Web エージェント FCC ページに対するクロスサイト スクリプティング攻撃を防ぐには、FCC 変数データが正しく表示されるように HTML エンコーディングを使用します。

HTML エンコーディングでは、文字が HTML 構文ではなく、そのリテラル値として扱われます。 エンコーディングにより、有害なクロス サイト スクリプティング構文は、表示するリテラル テキストとして描画され、HTML フォームの描画中にブラウザによってコードが実行されないようになります。 攻撃時に悪用される可能性のあるすべての構文をエンコードできます。

fcchtmlencoding パラメータは、以下の構文がある FCC 変数に挿入されたすべての値に HTML エンコーディング アルゴリズムを適用するようにエージェントに指示します。

```
$$varname$$
```

従来ブロックされている文字が FCC データで必要な場合は、fcchtmlencoding パラメータを有効にします。

fcchtmlencoding

Web エージェント FCC ページに対するクロスサイト スクリプティング攻撃を防ぐために、HTML エンコーディングを有効にするかどうかを指定します。 このパラメータは文字をブロックしません。

値： Yes および No

デフォルト： No

fcchtmlencoding パラメータは、すべての FCC フォームのすべての変数置換に適用されます。 このパラメータを使用するエージェントは、1 つ以上の FCC フォームに対応できます。

FCC ファイル内の特定の文字に HTML エンコーディングを適用するには、以下のパラメータを使用します。

fcchtmlencodingchars

特定の文字値を取得し、HTML エンコーディングを適用し、実際の値を FCC ファイル内のエンコードされた値に置き換えます。

重要: fcchtmlencodingchars パラメータを使用するには、fcchtmlencoding パラメータを「no」に設定する必要があります。

FCC ファイル内の特定の変数に HTML エンコーディングを適用するには、以下の関数を使用します。

HTMLENCODE

特定の変数値を取得し、HTML エンコーディングを適用し、実際の変数値を FCC ファイル内のエンコードされた値と置き換えます。

HTMLENCODE 関数の構文は、以下のとおりです。

```
$$htmlencode(varname)$$
```

重要: HTMLENCODE 関数を使用するには、fcchtmlencoding パラメータを No に設定する必要があります。

クロスサイト スクリプティングをチェックするための Web エージェントの設定

URL の中で、実行可能なスクリプトの一部になる可能性のある文字をチェックするよう Web エージェントに指示するには、CSSChecking パラメータを yes に設定します。 このパラメータを有効にすることにより、Web エージェントはクエリ文字列を含め URL 全体をスキャンします。そして、次のようなデフォルト文字セットのエスケープバージョンと、エスケープされていないバージョンの両方を探します。

- 左山形かっこと右山形かっこ（< および >）
- 引用符（'）

有効なターゲットドメインの定義

悪意のある Web サイトにユーザをリダイレクトするフィッシング攻撃からリソースを保護するように SiteMinder エージェントを設定するには、以下の設定パラメータを設定します。

ValidTargetDomain

認証情報コレクタがユーザをリダイレクトすることを許可されるドメインを指定します。 URL 内のドメインがこのパラメータ内で設定されたドメインに一致しない場合は、リダイレクトが拒否されます。

デフォルト： No

フォーム クレデンシャル コレクタ（FCC）を含むすべての高度な認証方式が、このパラメータをサポートします。

ValidTargetDomain パラメータは、処理の間にターゲットの有効なドメインを識別します。 ユーザをリダイレクトする前に、エージェントはリダイレクト URL の値と、このパラメータ内のドメインを比較します。 このパラメータがない場合、エージェントはユーザをどのようなドメイン内のターゲットにもリダイレクトさせます。

ValidTargetDomain パラメータには、有効なドメインごとに 1 つの値を設定し、複数の値を含めることもできます。

ローカル Web エージェント設定では、以下の例のように、各ドメインについて、1 行に 1 つのエントリを指定します。

```
validtargetdomain=".xyzcompany.com"

validtargetdomain=".abccompany.com"
```

クロスサイト スクリプティング攻撃からの J2EE アプリケーションの保護

攻撃者が要求に非標準（長過ぎる） Unicode 文字を使用することにより、クロス サイト スクリプティング保護をバイパスするのを防ぐことができます。

次の手順に従ってください:

1. CSSChecking パラメータの値を yes に設定します。
2. 以下のパラメータの値を yes に設定します。

   DisallowUTF8NonCanonical

   攻撃者が要求に非標準（長過ぎる） Unicode （utf-8）文字を使用することにより、クロス サイト スクリプティング保護をバイパスするのを防ぎます。 このパラメータの値が yes の場合、エージェントは非標準（長過ぎる）Unicode 文字が含まれる URL 要求をブロックします。

   デフォルト： No

CSS のデフォルト文字セットの上書き

デフォルトでは、エージェントは以下のデフォルトのクロス サイト スクリプティング文字セットを確認します。

- 左かっこと右かっこ：（、<、>、）
- アポストロフィ：'

このデフォルトの文字セットを使用しない場合は、独自の文字セットを定義する BadCSSChars エージェント パラメータを指定します。

BadCSSChars

指定された場合、デフォルトのクロス サイト スクリプティングの代わりに独自の文字セットを使用します。 希望する文字すべてからなる文字列全体を入力してください。

デフォルト：(<,',>)

例： <,> （この場合、エージェントは左山かっこと右山かっこのみをスキャンします。）

制限：

- これらの文字はそのまま指定できます。
- 最大 4096 文字まで指定できます（区切り文字として使用するカンマを含みます）。
- 文字の範囲をハイフンで区切って指定することもできます。構文は starting_character-ending_character です。 たとえば、一連の文字として「a-z」と入力できます。
- 引用符（"）を %22 の URL エンコード値で指定します。 ASCII は使用できません。

エージェントが文字セットに関する問題を検出すると、アクセス拒否メッセージをユーザに返し、エージェント エラー ログに以下のメッセージを記録します。

```
Caught Possible Cross Site Scripting Violation in URL. Exiting with HTTP 403 ACCESS FORBIDDEN.
```

一部のアプリケーションは、Web サーバ プラットフォームにかかわりなく、クエリ文字列の中で引用符を使用することを必要とします。 たとえば、iNotes Web Access のような特定の Domino アプリケーションは、引用符を使用することを必要とします。

クエリ文字列の中で引用符を必要とするアプリケーションを使用するには、BadCssChars パラメータから引用符を削除してください。

注: クロスサイト スクリプティングの詳細については、「CERT Advisory」を参照してください。

無効なクエリ文字の指定

URL のクエリ文字列部分に特定の文字を禁止するには、以下のパラメータを設定します。

BadQueryChars

Web エージェントによって、URL のクエリ文字列部分（'?' の後）で禁止される文字を指定します。

デフォルト： 空（クエリ文字列に禁止される文字はありません）

制限：

- デフォルトの 16 進数が英語の文字に適用されます。 その他の言語の場合、許可する言語の文字に対応する 16 進数値を削除します。 対象となる言語（ただしこれらに限定されません）には、ポルトガル語（ブラジル）、フランス語、日本語、および中国語などがあります。
- 文字は実際にその文字を入力して指定できます。 さらに、その文字の URL エンコード形式を入力することもできます。 たとえば、文字 a を入力するか、または、そのエンコード値である %61 を入力できます。
- 最大 4096 文字まで指定できます（区切り文字として使用するカンマを含みます）。
- 文字の範囲をハイフンで区切って指定することもできます。 構文は *starting_character-ending_character* です。 たとえば、一連の文字として「a-z」と入力できます。
- 引用符（"）を URL エンコード値 %22 で指定します。 ASCII は使用できません。

例： %25 は、クエリ内の URL エンコード文字をブロックします。

Web エージェントは、URL のクエリ文字列内の文字を BadQueryChars パラメータに定義された文字の ASCII 値と比較することにより、クエリ文字列に禁止された文字が含まれているかどうかを検索します。 たとえば、以下のように処理されます。

1. BadQueryChars パラメータに、以下に示すパーセント記号（%）の URL エンコード値が含まれているとします。

   %25

2. Web エージェントは、以下のクエリ文字列が含まれる HTTP リクエストを受信します。

   xxx=%0d

3. Web エージェントでは前述の例にある URL を検査しますが、URL エンコード値はデコードしません。 たとえば、Web エージェントによって前述の例（手順 2）がキャリッジ リターンではなくリテラル文字列 %0d として解釈されます。

4. Web エージェントは BadQueryChars パラメータの値を検索し、それらを ASCII 値に変換します。 たとえば、手順 1 の %25 はパーセント記号（%）に変換されます。

5. Web エージェントは、URL 内の各文字を、BadQueryChars パラメータからデコードされた ASCII 値と比較します。

6. その結果、ASCII パーセント記号（%）が以下の両方の場所に存在するため、Web エージェントによってリクエストがブロックされます。

   - URL のクエリ文字列
   - BadQueryChars パラメータのデコードされた（ASCII）値

クエリ文字列から特定の文字をブロックするには、BadQueryChars パラメータの値にブロックする文字を含めます。

無効な URL 文字の指定

URL リクエストの一部として使用できない一連の文字を指定することができます。 それらの文字は、エージェントにより、無効な URL 文字として扱われます。 このリスト内で指定されている文字または文字列を含む URL リクエストは、Web エージェントによって拒否されます。URL のうち、「?」文字より前の部分に対して、このチェックが実施されます。 悪意のある Web クライアントがそのような文字を使用して SiteMinder ルールを回避する場合があるので、Web エージェントはそのような文字が含まれる URL リクエストを拒否します。

Web エージェントが無効な URL 文字を含んでいる URL リクエストを拒否する場合、Web サーバは以下のメッセージのいずれかで応答します。

- 内部サーバ エラー
- Web ページが見つからないエラー（404）

エージェントが要求をどのように処理するかについては、Web エージェント ログを確認してください。

以下のパラメータを使用して文字を指定します。

BadUrlChars

URL リクエストに使用できない文字シーケンスを指定します。 Web エージェントはこのパラメータ内のリストに対して、「?」文字の前にある URL 内の文字を確認します。 指定された文字のいずれかが見つかった場合、Web エージェントは要求を拒否します。

以下の文字を指定できます。

- 円記号（¥）
- ダブルスラッシュ（//）
- ピリオドとスラッシュ（./）
- スラッシュとピリオド（/.）
- スラッシュとアスタリスク（/*）
- アスタリスクとピリオド（*.）
- ティルダ（~）
- %2d
- %20

- %00-%1f
- %7f-%ff
- %25

複数の値はカンマで区切ります。 スペースは*使用しないでください*。

無効な URL 文字は、その前に疑問符 (?) が付いている場合にのみ、CGI パラメータの中で使用できます。

デフォルト： 無効（すべての文字が許可されます）。

制限：

- デフォルトの 16 進数が英語の文字に適用されます。 その他の言語の場合、許可する言語の文字に対応する 16 進数値を削除します。 対象となる言語（ただしこれらに限定されません）には、ポルトガル語（ブラジル）、フランス語、日本語、および中国語などがあります。
- 文字は実際にその文字を入力して指定できます。 さらに、その文字の URL エンコード形式を入力することもできます。たとえば、文字 a を入力するか、または、そのエンコード値である %61 を入力できます。
- 最大 4096 文字まで指定できます（区切り文字として使用するカンマを含みます）。
- 文字の範囲をハイフンで区切って指定することもできます。構文は *starting_character-ending_character* です。 たとえば、一連の文字として「a-z」と入力できます。
- 引用符（"）を URL エンコード値 %22 で指定します。 ASCII は使用できません。

無効な URL 文字を指定するには、BadURLChars パラメータの値をブロックする文字が含まれるように編集します。

注: Apache 2.0 リバース プロキシ サーバおよび Outlook Web Access（OWA）を設定する際は、必ず BadURLChars パラメータをオフにしてください。 OWA では、電子メール件名中の文字が、BadURLChars パラメータでリストされていたとしても、無制限に許可されてしまうからです。

不正な形式の文字の有効化

クロスサイト スクリプティング攻撃では一般に以下の文字が使用されます。

- 左山形かっこと右山形かっこ（< と >）
- アンパサンド（&）
- 引用符（"）

認証のチャレンジ中に、ユーザへのフォームの表示にスクリプト コードを使用する場合は、以下のパラメータを有効にして、任意の特殊文字を HTML フォームに送信する前に、それらをブロックするように Web エージェントを設定します。

BadFormChars

フォーム上で出力としてそれらを使用する前に、Web エージェントでブロックする文字を指定します。有効かつ URL のエージェント名部分にこのパラメータで指定される 1 つ以上の文字がある場合、ログイン ページは以下のエラー メッセージを返します。

内部サーバ エラー

デフォルト： 無効（文字はブロックされません）

例： <、>、&、%22

制限：

- これらの文字はそのまま指定できます。
- 最大 4096 文字まで指定できます（区切り文字として使用するカンマを含みます）。
- 文字の範囲をハイフンで区切って指定することもできます。構文は starting_character-ending_character です。 たとえば、一連の文字として「a-z」と入力できます。
- 引用符（"）を %22 の URL エンコード値で指定します。 ASCII は使用できません。

以下の手順に従います。

1. 管理 UI にログインします。
2. このパラメータを有効にするエージェント設定オブジェクトを開きます。

3. BadFormChars パラメータを先頭の # 文字を削除することによって有効にします。

   BadFormChars パラメータは、デフォルト値で有効になっています。

4. （任意）使用しないすべての文字をリストから削除します。 他の文字もこのリストに追加できます。 文字が互いにカンマで区切られていることを確認します。

DNS サービス拒否攻撃の防御支援

IP アドレスに誤りのある有効な HTTP 要求を Web サーバが受信した場合、Web エージェントはその IP アドレスを完全修飾ドメイン名に解決しようとします。HTTP 要求のボリュームが大きい場合、サービス妨害状態が Web エージェントと、場合によっては DNS サーバに影響を与える可能性があります。 以下のパラメータは、Web エージェントが DNS 参照を実行するかどうかを制御します。

DisableDNSLookups

Web エージェントが DNS の検索を実行することを防ぎます。

次の手順に従ってください:

1. DisableDNSLookup パラメータが s で終了 *しない*ことを確認します。 ACO テンプレートおよび LocalConfig.conf ファイルの旧バージョンの一部にはこのエラーが含まれている可能性があります。 正しいパラメータは p で終了します。
2. DisableDNSLookup パラメータの値を yes に設定します。

重要: このパラメータの値が yes に設定された場合、Cookie ベースの機能が正しく動作するためには完全修飾ドメイン名が必要です。

拡張子のないリソースの保護

不正なユーザが拡張子のないリソースへのアクセスを取得するのを防ぐために、以下のパラメータを使用することができます。

OverrideIgnoreExtFilter

Web エージェントがすべての URI と比較する目的で使用するために、複数の文字列から成る 1 つのリストを指定します。 これは、通常は拡張子が Web エージェントによって無視されるリソース、または拡張子のないファイルやアプリケーションの保護に役立ちます。 URI がリスト内の文字列の 1 つと一致する場合、Web エージェントはポリシー サーバへの問い合わせを行い、そのリソースが保護されているかどうかを決定します。

パスを厳密に指定するのではなく一般的な文字列を指定することをお勧めします。 一群のリソースを保護するために部分的な文字列を含めることもできます。 たとえば、指定された文字列が /servlet/ の場合、以下のリソースが保護されます。

- /dira/app1/servlet/app
- /dirb/servlet/app1
- /dirc/mydir/servlet/app2

デフォルト： デフォルトなし

拡張子のないリソースを保護するには、OverrideIgnoreExtFilter パラメータの値に保護するリソース（期間のない）の文字列を追加します。エージェント設定オブジェクトを使用している場合は、文字列を追加するために複数値オプションを使用します。 ローカル設定ファイルを使用している場合は、各文字列を個別の行に追加します。

POST 維持の無効化

POST 維持を使用する必要がない場合は、以下のパラメータを使用してそれを無効にすることができます。

PreservePostData

要求をリダイレクトする場合に Web エージェントが POST データを維持するかどうかを指定します。 ユーザが、フォーム認証や証明書認証など、高度な認証を受ける場合、POST データは認証フェーズの間、維持されます。

デフォルト： yes

POST 維持を無効にするには、PreservePostData パラメータの値を no に設定します。

アプリケーションのセキュリティ保護

無許可のユーザは、Web エージェントが無視するように設定されている拡張子を含む虚偽のファイル名を URL の最後に追加することができます。追加すると、無許可のユーザがそのリソースにアクセスできるようになります。 そのような試行へのアクセスを Web エージェントに拒否させるには、以下のパラメータを使用します。

SecureApps

エージェントが、権限のないユーザからの URL を許可することを防ぎます。 Web エージェントが、特定の拡張子で終わるファイルに対するリクエストを無視するように設定されている場合は、偽の URL を作成してリソースにアクセスしようとする攻撃を受ける可能性があります。

たとえば、以下の URL を持つリソースがあるとします。

```
/scripts/myapp
```

以下の例のような偽の URL を作成して、アクセス権を取得しようとする攻撃を受ける可能性があります。

```
/scripts/myapp/junk.jpg
```

SecureApps パラメータの値が no の場合に、Web エージェントが .jpg ファイルのリクエストを無視するように設定されていると、/scripts/myapp/junk.jpg のリクエストは自動的に許可されます。

SecureApps パラメータの値が yes の場合は、Web エージェントは、リソースが正当であるか、URL が偽であるかの検出を試みます。

デフォルト： No

アプリケーションを保護するには、SecureApps パラメータの値を yes に設定します。

カスタム レスポンスの X-Frame Options への準拠の確認

Web アプリケーションで X-Frame-Options レスポンス ヘッダを使用すると、エージェントからのカスタマイズされたレスポンスがこのヘッダを正しく返すことを確認できます。 X-frame-options ヘッダ内の設定によって、ブラウザが <frame> または <iframe> タグの間のコンテンツを持ったページを表示するかどうかが決まります。

以下のパラメータを使用して、エージェントからのカスタム レスポンスが X-frame-options に準拠するかどうかを決定できます。

XFrameOptions

カスタム レスポンスが x-frame-options レスポンス ヘッダに準拠するかどうかを指定します。 このパラメータを設定すると、正しい x-frame-options ヘッダでカスタム レスポンスが設定されます。

デフォルト： No（レスポンスは x-frame options レスポンス ヘッダに準拠しません）

範囲： Yes、No

カスタム レスポンスが x-frame-options に準拠していることを確認するには、XFrameOptions パラメータの値を yes に設定します。

IP アドレスの確認

SiteMinder エージェントは以下の手順で解説されているパラメータを使用して、IP アドレスを確認できます。

- IP アドレスによってエージェント ID を解決します (P. 103)。
- セキュリティ侵害を防止するために IP アドレスを比較します (P. 104)。

IP アドレスによるエージェント ID の解決

仮想 Web サーバ上で、IP アドレスとホスト名を使用してエージェント名を解決する場合、Web エージェントは誤った AgentName の値を使用してリクエストを評価することがあります。 これにより、認証されていないユーザが保護されたリソースにアクセスする状況が生じることがあります。

以下のパラメータを使用して、Web エージェントが仮想サーバの物理 IP アドレスに基づいてエージェント名を解決するように設定することができます。

UseServerRequestIp

仮想 Web サーバの物理 IP アドレスに従って AgentName を解決するように Web エージェントに指示します。Web サーバが仮想サーバ マッピングに IP アドレスを使用する場合は、このパラメータを使用してセキュリティを向上させます。このパラメータが no の場合、Web エージェントは、クライアントのリクエストの HTTP ホスト ヘッダ内のホスト名に従って AgentName を解決します。

Domino サーバでは、このパラメータは、Domino 6.x でのみサポートされています。他の Domino バージョン上のエージェントに対してこのパラメータを有効にすると、Web エージェントはデフォルトのエージェント名を使用します。

SSL 通信と仮想ホストを使用するように IIS Web エージェントが設定されている場合は、このパラメータを yes に設定する必要があります。 IIS では、SSL を有効にした状態で、ホスト名を使用して仮想ホスト マッピングを行うことはできません。

デフォルト： No

IP アドレスを使用して Web エージェントの ID を解決するには、UseServerRequestIp パラメータを yes に設定します。

セキュリティ侵害を防止するための IP アドレスの比較

無許可のシステムでは、パケットを監視して Cookie を不正に入手し、その Cookie を使って別のシステムにアクセスすることができます。 無許可のシステムによるセキュリティ侵害を防止するために、永続的な Cookie と一時的な Cookie を使用して、IP チェックを有効または無効にすることができます。

IP チェック機能では、エージェントが現在の要求に含まれている IP アドレスに対して最後の要求の Cookie に格納された IP アドレスを比較する必要があります。 IP アドレスが一致しない場合、エージェントは要求を拒否します。

IP チェックを実装する目的で使用される 2 つのパラメータは、PersistentIPCheck と TransientIPCheck です。これらを次のように設定します。

- PersistentCookies を有効にした場合、PersistentIPCheck を yes に設定します。
- PersistentCookies を有効にしなかった場合、TransientIPCheck を yes に設定します。

SiteMinder ID Cookie は、IP チェックの影響を受けません。

詳細情報

永続的 Cookie の設定 (P. 109)
識別 Cookie の制御 (P. 108)

SiteMinder ブラウザ Cookie

SiteMinder エージェントと関連付けられる Cookie を管理するには、以下の手順でパラメータを使用します。

- 基本認証方式に Cookie が必要です (P. 106)。
- HTTP 専用属性による Cookie 情報を保護します (P. 107)。
- ドメイン内で安全な Cookie を設定します (P. 107)。
- アイデンティティ Cookie を制御します (P. 108)。
- 永続的 Cookie を設定します (P. 109)。
- エージェント Cookie の Cookie パスを指定します (P. 110)。
- エージェントに Cookie ドメインの使用を強制します (P. 112)。
- Cookie ドメイン解決を実行します (P. 113)。
- Cookie パス範囲設定の仕組み (P. 114)。

基本認証用の Cookie が必要です。

以下のパラメータを使用して、SiteMinder が Cookie を必要とするかどうかを制御できます。

RequireCookies

SiteMinder が Cookie を必要とするかどうかを指定します。SiteMinder では、以下の機能のために Cookie が必要です。

- シングル サインオン環境の保護。
- セッション タイムアウトの適用。
- アイドル タイムアウトの適用。

このパラメータの値が Yes の場合、エージェントは HTTP 要求を処理するために以下のいずれかの Cookie を必要とします。

- SMCHALLENGE
- SMSESSION

このパラメータの値が No の場合、次の条件が発生する可能性があります。

- ユーザに対して、予期せず認証情報の認証が行われます。
- タイムアウトが厳密に適用されません。

重要: エージェントが Cookie を必要とする場合は、ブラウザ内で HTTP Cookie を受け入れるようにユーザに指示してください。そうしない場合、ユーザはすべての保護リソースへのアクセスを拒否されます。

デフォルト： Yes

Cookie を必要とするには、RequireCookies パラメータの値を yes に設定します。

HTTP 専用属性を備えた Cookie での情報の保護

クロスサイト スクリプティング攻撃に対する保護に役立つように、Web エージェントに、次のパラメータを使用して、作成するすべての cookie に HTTP 専用属性を設定させることができます。

UseHTTPOnlyCookies

Web エージェントが作成する cookie で HTTP のみの属性を設定するように Web エージェントに指示します。 Web エージェントが、ユーザのブラウザにこの属性を持つ cookie を返すと、その cookie の内容はスクリプトで読み取ることができなくなります。これは、cookie を最初に設定した Web サイトのスクリプトにも当てはまります。これにより、cookie 内の機密情報を、権限のないサードパーティがスクリプトを使用して読み取ることを防ぐことができます。

デフォルト： No

cookie 内の情報を保護するには、UseHTTPOnlyCookies パラメータの値を yes に設定します。

安全な Cookie の設定

以下のパラメータを使用して、安全な（HTTPS）接続上の保護されている Web サーバとリ要求ブラウザの間でのみセッション cookie が送信されるように指定することができます。

UseSecureCookies

安全な（HTTPS）接続を使用して、Web サーバに cookie を送信します。 このパラメータを使用することで、ブラウザと Web サーバの間のセキュリティを向上させることができます。

この設定が有効な場合、シングル サインオン環境のユーザは、SSL Web サーバから非 SSL Web サーバに移動するときに再認証する必要があります。セキュア cookie は、従来の HTTP 接続を介して渡すことはできません。

デフォルト： No

SSL 接続経由で cookie を送信するには、UseSecureCookies パラメータを yes に設定します。

詳細情報:

複数ドメインにわたる安全な Cookie の設定 (P. 288)

識別 Cookie の制御

TransientIDCookies パラメータは、エージェント ID cookie（SMIDENTITY）が一時的か永続的かを指定します。

永続的 cookie は、クライアント システムのハード ディスクに書き込まれます。 Web エージェント 5.x QMR1 より前のバージョンでは、永続的な cookie は 7 日間にわたって有効でした。 Web エージェント 5.x QMR1 以降では、永続的な cookie は、設定済みの最大セッションタイムアウト + 7 日間にわたって有効です（最大セッション タイムアウトの設定には 管理 UI を使用します）。通常、有効期限を過ぎると永続的な cookie は Web ブラウザの cookie ファイルから削除されます。ただし、永続的な cookie の処理方法は、ブラウザによって異なります。デフォルトでは、Web エージェントは永続的な cookie を使用しません。 Web エージェントは、過渡的な cookie を使用します。

複数のブラウザセッションでシングル サインオンを使用するには、永続的な Cookie を使用します。 永続的な cookie を設定すると、ユーザは、SiteMinder セッションの有効期限が切れる前にブラウザ セッションを終了できます。新しいブラウザ セッションを開始しても、シングル サインオン機能は有効のままです。

永続的な cookie がハードディスクに書き込まれるのに対し、過渡的な cookie はハードディスクに書き込まれることはなく、設定済みのセッションタイムアウトの対象になりません。過渡的な cookie は、使用中の cookie フォルダ内にとどまります。

ID cookie を永続的なものにしたい場合、TransientIDCookies を no に設定します。 ID cookie を一時的なものにしたい場合、デフォルト値である yes のままにしておきます。

対応する IP チェック機能を確実に設定してください。

詳細情報

セキュリティ侵害を防止するための IP アドレスの比較 (P. 104)

永続的 Cookie の設定

複数のブラウザセッションでシングル サインオンを使用するには、永続的な Cookie を使用します。 以下の手順では、永続的な Cookie の考えられる 1 つの使用法について説明します。

1. ユーザは SiteMinder で認証しますが、SiteMinder セッションが期限切れになる前に、ブラウザ セッションを終了します。
2. ユーザは後で新規ブラウザ セッションを開始しますが、永続的な Cookie はシングル サインオン機能を維持します。

永続的な Cookie は、設定された最大セッション タイムアウトに 7 日間*加えた*期間で有効です。Cookie が期限切れになった後、多くのブラウザは、Web ブラウザの Cookie ファイルを削除します。 一部のブラウザは永続的な Cookie を異なる方法で処理する場合があります。

次の手順に従ってください:

1. PersistentCookies パラメータに yes を設定します。

   SMSESSION Cookie は永続的です。

2. TransientIDCookies パラメータを no に設定します。

   SMIDENTITY Cookie は永続的です。

エージェント Cookie の Cookie パスの指定

Web エージェントが cookie を作成する場合、Web エージェントは自動的に cookie パスとしてルート（/）ディレクトリを使用します。 cookie のドメインとパスの属性は要求の URL と比較されます。cookie がドメインとパスに対して有効な場合、クライアントはサーバに cookie を送ります。 cookie パスがルート値を使用する場合、クライアントはドメインのすべての要求を備えたサーバに cookie を送ります。

指定された一連のパスに SiteMinder cookie を設定すると、保護されていないリソースに cookie が送信されるときに生じる Web トラフィックを除去できます。 たとえば、cookie パスを /mypackage に設定すると、クライアントはドメインの特定のパッケージ内の要求に対してのみ、cookie を送ります。

エージェント cookie の cookie パスを指定する方法

1. エージェント設定オブジェクトまたはローカル エージェント設定ファイルを開きます。
2. 以下のパラメータ内で cookie プロバイダの cookie パスを設定します。

   MasterCookiePath

   cookie プロバイダによって作成されたプライマリ ドメイン セッション cookie のパスを指定します。 たとえば、このパラメータが /siteminderagent に設定されている場合、cookie プロバイダが作成するすべてのセッション cookie のパスに /siteminderagent が含まれます。 cookie プロバイダ エージェントにこのパラメータが設定されていない場合は、デフォルト値が使用されます。

   デフォルト： /（ルート）
3. 以下のパラメータ内でセカンダリ エージェントの cookie パスを設定します。

   CookiePath

   以下のセカンダリ エージェント ブラウザ cookie の cookie パスを指定します。

   - xxSESSION
   - xxIDENTITY
   - xxDOMINODATA
   - xxCHALLENGE（SSL_CHALLENGE_DONE を含む）
   - xxDATA

- xxSAVEDSESSION

たとえば、このパラメータを /BasicAuth に設定すると、前述のリスト内のセカンダリ エージェントはすべて、パスとして /BasicAuth を使用して作成されます。 指定しない場合は、デフォルト値が使用されます。

4.x のエージェントとの後方互換性を維持するため、CookiePath は認証情報 cookie（xxxxCRED など）には追加 *されません*。

以下の cookie は *常に*ルート パス（/）を使用します。

- ONDENIEDREDIR
- TRYNO

CookiePathScope パラメータが 0 より大きい場合は、CookiePath パラメータの設定が上書きされます。

デフォルト： /（ルート）

4. （オプション）CookiePath 値を使用する代わりに Web エージェントに URL から cookie パスを抽出させたい場合、以下のパラメータを 0 を超える数に設定します。

CookiePathScope

以下のセカンダリ エージェント cookie の cookie パスの範囲を指定します。

- xxSESSION
- xxIDENTITY
- xxDOMINODATA
- xxCHALLENGE（SSL_CHALLENGE_DONE を含む）
- xxDATA
- xxSAVEDSESSION

CookiePathScope が 0 より大きい場合は、CookiePath パラメータの設定が上書きされます。

デフォルト： 0

詳細情報:

完全ログオフの設定 (P. 296)

Cookie ドメインの強制

完全修飾ドメイン名を使用すると、Cookie が正しく動作することを保証できます。 以下のいずれかの条件を満たす URL 要求内のホスト名の末尾に Cookie ドメインを追加するようにエージェントに強制することができます。

- 要求でドメインが指定されていない
- 要求に IP アドレスのみが含まれる
- 以下のパラメータの設定により、強制的にエージェントが Cookie ドメインを使用するようにできます。

  ForceCookieDomain

  ドメインが指定されていない URL リクエスト内や IP アドレスのみで構成されている URL リクエスト内のホスト名に cookie ドメインを追加するように Web エージェントに強制します。 このパラメータを ForceFQHost パラメータと組み合わせると、機能を追加できます。

  デフォルト： No

  ForceFQHost

  完全修飾ドメイン名を使用するようにエージェントに強制します。このパラメータは設定されたドメイン ネーム システム (DNS) サービスを使用して、URL 要求内のホスト名に Cookie ドメインを追加します。このとき、エージェントではなく DNS サービスが使用されます。 Web エージェントは、URL の一部が含まれた要求を受信すると、元の URI に指定されている同じ転送先リソースに、その要求をリダイレクトします。 リダイレクト要求では、完全修飾ホスト名が使用されます。完全修飾ホスト名は、設定されている DNS サービスを使用してエージェントが決定するものです。 このパラメータを ForceCookieDomain パラメータと組み合わせて使用すると、機能を追加できます。

  デフォルト： No

  例：エージェントは http://host1/page.html から要求を受け取ると、http://host1.myorg.com/page.html で応答します。このエージェントが http://123.113.12.1/page.html などの要求を受け取ると、http://host1.myorg.com/page.html で応答します。

  注：これらの例は適切な DNS ルックアップ表でのみ動作します。DNS が解決できない部分的なドメイン名が含まれる要求は、エラーを生成する場合があります。

次の手順に従ってください:

1. ForceCookieDomain パラメータの値を yes に設定します。
2. ForceFQHost パラメータの値を yes に設定します。

   必要に応じてエージェントはホスト名の末尾にその Cookie ドメインを追加します。

Cookie ドメイン解決の実装

自動的なドメイン解決を実装するには、CookieDomain パラメータをコメントアウトするか、そのパラメータを none に設定して、発行元のサーバ上でのみ有効な cookie を作成するよう Web エージェントに指示します。

さらに、CookieDomainScope パラメータに値を追加して、cookie ドメインを定義することもできます。 有効範囲には、ドメイン名を構成するセクションの数を、ピリオドで区切って指定します （ドメインは必ず「.」で始まります）。

CookieDomainScope の値が 0 である場合、特定のホストに対して最も具体的な有効範囲を使用するようエージェントに指示します。 1 という値（たとえば、.com という cookie ドメインを生じさせます）は、HTTP 仕様により許可されていません。 2 という値は、最も一般的な有効範囲を使用するようエージェントに指示します。

以下の表に、ドメイン名と CookieDomainScope の値をいくつか示します。

ドメイン名	cookie ドメインの有効範囲の値	cookie ドメイン
server.myorg.com	2	.myorg.com
server.division.myorg.com	3	.division.myorg.com
	2	.myorg.com
server.subdivision.division.myorg.com	4	.subdivision.division.myorg.com
	3	.division.myorg.com
	2	.myorg.com

たとえば、division.myorg.com ドメインの有効範囲は 3 です。 デフォルトでは、Web エージェントは有効範囲として 2 を想定しています。cookie ドメインの有効範囲を 1 にすることはできません。

CookiePathScope 設定の機能

以下の表は、CookiePathScope パラメータの値が以下の設定でどのように機能するかを示しています。

- http://*fqdn*/path1/path2/path3/path4/index.html のような URL
- /BasicA の CookiePath パラメータ値

CookiePath 値	CookiePathScope 値	使用されるパス
/BasicA	0	/BasicA
/BasicA	1	/Path1
/BasicA	2	/Path1/Path2
/BasicA	3	/Path1/Path2/Path3
/BasicA	4	/Path1/Path2/Path3/Path4
/BasicA	5	/Path1/Path2/Path3/Path4
/BasicA	99	/Path1/Path2/Path3/Path4
/または「未定義」	0	/
/または「未定義」	1	/Path1
/または「未定義」	2	/Path1/Path2
/または「未定義」	3	/Path1/Path2/Path3
/または「未定義」	4	/Path1/Path2/Path3/Path4
/または「未定義」	5	/Path1/Path2/Path3/Path4
/または「未定義」	99	/Path1/Path2/Path3/Path4

これらの設定はさらに単純な SSO にも影響します。 たとえば、CookiePathScope の値が 1 以上に設定されている場合、パスが/BasicA のセッション cookie が/BasicB/Index.html リクエストで有効にならないので、ユーザは/BasicA/Index.html と/BasicB/Index.html の両方の認証情報を要求されます。

SDK サードパーティ Cookie のサポート

組織で SiteMinder 以外の Web エージェントを使用する場合は、以下のパラメータを使用して、シングル サインオンがサポートされるようにそれらの Web エージェントを設定できます。

AcceptTPCookie

Web エージェントがサードパーティ（SiteMinder 以外）の Web エージェントによって作成されたセッション（SMSESSION）cookie を受け取ることを可能にします。 サードパーティ エージェントは、SiteMinderSDK を使用して、SMSESSION cookie を生成したり読み取ったりします。

デフォルト： デフォルトなし

注: 詳細については、「SiteMinder Developer's Guides」を参照してください。

Web エージェントが SiteMinder 以外の Web エージェントによって作成されたセッション cookie を受け入れることができるようにするには、AcceptTPCookie パラメータを yes に設定します。

HTTPS ポートの定義

要求をより安全にしておくために Web サーバ（HTTPS）への SSL 接続を使用している場合は、以下のパラメータを使用して HTTPS ポート番号を指定します。

HttpsPorts

ユーザが Web サーバへの SSL 接続を使用しているかどうかを Web エージェントがリスンする安全なポートを指定します。 このパラメータの値を指定する場合、安全な要求を提供するすべての Web サーバの対象となるすべてのポートを含める必要があります。 値を指定しなかった場合、Web エージェントは HTTP スキームをサーバのコンテキストから読み取ります。

サーバが、（HTTPS を HTTP へ変換する） HTTPS アクセラレータの背後にある場合、ブラウザはその要求を SSL 接続として扱います。

デフォルト： 空白

例： 80

例： （複数のポート） 80,8080,8083

HTTPS ポートを定義するには、HttpsPorts パラメータの値を SSL を使用するポート番号に設定します。 ポート番号が複数ある場合は、カンマで区切ります。

URL 内のクエリ データのデコード

ポリシー サーバを呼び出す前に、Web エージェントの Base64 アルゴリズムが URL のクエリ データをデコードするように設定する（それによってポリシー サーバは適切なリソースを参照します）には、以下のパラメータを使用します。

DecodeQueryData

ポリシー サーバをコールする前に、Web エージェントが URL 内のクエリ データをデコードするかどうかを指定します。 環境内で以下のタスクのいずれかを実行する必要がある場合は、このパラメータを yes に設定します。

- 正しい文字列に対してルール ファイラが機能していることを保証する必要がある場合
- クエリ文字列内のデータに対して書き込みルールが機能していることを保証する必要がある場合

デフォルト： No

ポリシー サーバをコールする前に、Web エージェントが URL のクエリ データをデコードするように設定するには、DecodeQueryData パラメータの値を yes に設定します。

期間や拡張のないリソースを保護する方法

サーブレットなど、期間のない URL があります。 拡張のない URL もあります。 これらの状況は両方ともセキュリティ リスクをもたらします。 以下の手順で、これらのリスクを実証します。

1. 使用している環境には、保護されたリソースである、/mydir/servlets というディレクトリが含まれています。
2. Web エージェントは拡張子が .gif のリソースに対する要求を無視するように設定されています。
3. 不正なユーザが、以下の例に示されるように URL の最後に拡張子 .gif と共に架空のファイルの名前を追加します。

   /mydir/servlets/file.gif
4. Web エージェントは拡張子 .gif を無視し、不正なユーザに /mydir/servelets ディレクトリへのアクセス権を与えます。

セキュリティのリスク回避が最優先事項である場合、エージェントがどの拡張子も無視できないようにしてください。その際、次のような結果が生じることを考慮してください。

- Web エージェントがページ上にあるイメージの URL をすべて評価するので、パフォーマンスが低下することがあります。
- 以前は認証が不要だったリソースに関してユーザが認証を要求されることがあるため、Web サイトの動作が変わることがあります。

期間がない URL を保護するには、以下のオプションがあります。

- OverrideIgnoreExtFilter 機能を使用するよう、エージェントを設定します。
- 保護されているリソースに、Web エージェントが無視するよう設定されている拡張子が付いていないことを確認します。

複雑な URI の処理

DisableDotDotRule パラメータは、Web エージェントが、スラッシュ（/）で区切られた 2 つのドットを含む URI を自動的に許可するかどうかを判別します。

デフォルト： No

DisableDotDotRule が yes に設定されている場合、エージェントは二重ドット ルールを*適用しません*。 たとえば、以下の URI を考えます。

- /dir1/app.pl/file1.gif

  Web エージェントは、IgnoreExt パラメータを使用して、リソースを自動許可するかどうかを判別します。

- /dir1/okay.button.gif

  エージェントはこの URI を無視できます。これは、2 つのドットはスラッシュ（/）で区切られていないためです。 二重ドット ルールは、この場合は適用されません。

DisableDotDotRule が no に設定されていると（デフォルト）、Web エージェントは二重ドット ルールを*適用します*。 Web エージェントは以下の URI に対する要求の認証を要求し、要求をポリシー サーバに渡します。

- /dir1/app.pl/file1.gif

  この URI は二重ドット ルールに当てはまります。これは、2 つのドットはスラッシュで区切られているためです。

  Web サーバは、/dir1/app.pl をターゲット リソースと見なし、/file1.gif を追加のパス情報と見なすことができます（一般に、このパス情報は、CGI ヘッダで PATH_INFO として表示可能です）。

- /dir1/okay.button.gif

  エージェントはこの URI を無視することができます。これは、二重ドット ルールは実施されているのに、2 つのドットがスラッシュ（/）で区切られていないため、ルールが適用されないからです。

重要: 許可されていないアクセスが発生することのないように、IgnoreExt パラメータおよび DisableDotDotRule パラメータは併用しないでください。たとえば、/dir1/app.pl の保護が必要であるにもかかわらず、DisableDotDotRule パラメータを yes に設定すると、エージェントは URI /dir1/app.pl/file1.gif を無視します。これは、二重ドット ルールを無効にし、.gif を IgnoreExt パラメータに含めたためです。 その結果、許可されていないユーザが保護されたアプリケーション /dir1/app.pl にアクセスできるようになります。

第 7 章: SiteMinder エージェントでのプライバシー優先プロジェクト用のプラットフォーム(P3P)のコンパクト ポリシーの使用

SiteMinder エージェントは以下の手順で解説されているパラメータを使用して、P3P コンパクト ポリシーをサポートできます。

- P3P コンパクト ポリシーをサポートする方法 (P. 121)。
- P3P コンパクト ポリシーをサポートするように、エージェントを設定します (P. 122)。

このセクションには、以下のトピックが含まれています。

SiteMinderWeb エージェントで P3P コンパクト ポリシーをサポートする方法

CA SiteMinder は、以下のタイプを*除いて*、ほとんどのタイプの Web エージェントで P3P コンパクト ポリシーをサポートします。

- Domino

注: P3P の詳細については、World Wide Web Consortium の Web サイトの P3P のページを参照してください。

P3P コンパクト ポリシーをサポートするように Web エージェントを設定するには、以下の手順に従います。

1. Web サーバで P3P コンパクト ポリシーを設定します。

   注: 詳細については、Web サーバ ベンダーによって提供されているマニュアルを参照してください。

2. P3P コンパクト ポリシーに対応するように Web エージェントを設定します。

Web エージェントの設定による P3P コンパクト ポリシーへの対応

以下のパラメータを使用して、Web エージェントからのカスタム レスポンスが P3P レスポンス ヘッダに準拠するかどうかを決定できます。

P3PCompactPolicy

カスタム レスポンスがプライバシ優先プロジェクト用のプラットフォーム（P3P）レスポンス ヘッダに準拠するかどうかを決定します。P3P コンパクトポリシーは、P3P の用語に基づく特定の要素を表すトークンを使用します。 P3PCompactPolicy パラメータを適切なポリシー構文に設定した場合、Web エージェントに関して P3P レスポンス ヘッダが指定されている状況で、正しい P3P レスポンス ヘッダを使用して、カスタム レスポンスが設定されることを保証できます。

デフォルト： デフォルトなし

例： NON DSP COR CURa TAI （これらはそれぞれ、none、disputes、correct、current/always、および tailoring を表します）

P3P コンパクト ポリシーに対応するには、P3PCompactPolicy パラメータに適切なポリシー構文を追加します。

第 8 章: セッション保護

このセクションには、以下のトピックが含まれています。

Web アプリケーション クライアントへの SiteMinder 動作の適用

一部の Web アプリケーションでは、リソースをリクエストし、コンテンツを表示するために Web ブラウザのコンテキストで実行するスクリプト エンジンを使用します。 標準的な Web ブラウザが送信するリクエストと同様に、スクリプト エンジンから送信されたリクエストは、HTTP リダイレクトやチャレンジなどの SiteMinder で生成された動作をトリガできます。

Web アプリケーションに適切に統合されていないと、この動作により Web アプリケーション クライアントが不確定状態になる可能性があります。

Web アプリケーション クライアント レスポンス（WebAppClientResponse）ACO パラメータを使用して、以下の操作を実行できます。

- Web ブラウザのコンテキストで実行しているスクリプト エンジンから送信されたリクエストを識別するように SiteMinder を設定する。
- チャレンジなどの SiteMinder で生成された動作を、Web アプリケーション クライアントの機能と統合するためにカスタマイズしたレスポンスを使用する。

  注: SiteMinder のセッション管理機能（アイドル タイムアウトまたはセッション タイムアウトなど）を統合するために WebAppClientResponse パラメータを使用している場合は、OverLookSessionFor ACO パラメータも設定します。

  OverLookSessionFor パラメータは Web アプリケーション クライアント リクエストによってユーザ セッションが無限にアクティブにならないようにしますが、WebAppClientResponse パラメータでは、セッション タイムアウト後にユーザをリダイレクトするための必須 SiteMinder 機能を統合できます。

詳細情報:

セッション タイムアウト後のユーザのリダイレクト (P. 140)
メソッドと URI に基づいたセッション Cookie の作成または更新の防止 (P. 137)

Web アプリケーション クライアント レスポンスの導入

WebAppClientResponse ACO パラメータを使用して、SiteMinder セキュリティを維持しながら Web アプリケーション クライアントの機能を実装します。

パラメータは以下のデフォルト属性で構成されます。

```
Resource=|Method=|Status=|Body=|ContentType=|Charset=
```

以下の点を考慮してください。

- この ACO パラメータは、有効な値を持つ 1 つ以上の属性を必要とします。
- 追加の属性はすべてオプションです。
- 複数の Web アプリケーションからのリクエストを識別する必要がある場合、単一の ACO パラメータには属性ごとに複数の値を含めることができます。
- Web アプリケーション クライアント レスポンス機能は、基本認証方式では動作しません。

例: WebAppClientResponse ACO パラメータ

この例では、各属性に対して有効な値を持つパラメータを示します。 例の後に各属性の説明が示されます。

```
WebAppClientResponse:Resource=/web20/dir/*|Method=GET,POST|Status=200

|Body=C:¥location¥custombody_1.txt|Content-Type=application/xml|Charset=us-ascii
```

Resource

Web アプリケーション クライアントがリクエストを行う URI を指定します。リクエストの URI がこの値に一致する場合、SiteMinder は Web アプリケーション クライアントから送信されたものとしてリクエストを識別します。 リソースには、プレフィックスとサフィックスをマッチングするためにワイルドカード（*）を含めることができます。

デフォルト： 値なし。 この値を省略した場合、Web エージェントが保護しているすべてのリソースがパラメータに適用されます。

制限： 正規表現はサポートされていません。

例： Resource=/web20/dir/*

例： Resource=/web20/dir/*.xml

Method

Web アプリケーション クライアントがリクエストを行うための HTTP メソッドを指定します。 リクエストの HTTP メソッドがこの値に一致する場合、SiteMinder は Web アプリケーション クライアントから送信されたものとしてリクエストを識別します。

デフォルト： 値なし。 この値を省略した場合、パラメータは HTTP メソッドをすべて適用します。

複数のメソッドは、カンマ（,）で区切ります。

例： GET, POST

Status

SiteMinder が Web アプリケーション クライアント リクエストに送り返す必要がある HTTP ステータスを指定します。

デフォルト： 値なし。 この値を省略した場合、200 の HTTP ステータスがパラメータに適用されます。

Body

Web アプリケーション クライアント リクエストに対するレスポンスとして機能するカスタム本文が含まれるファイルの完全修飾名を指定します。 このファイルは Web エージェント ホスト システム上に存在し、以下のように指定できます。

- テキストベースにするか、バイナリ データを含める。
- アプリケーションの所有者が設計した任意のカスタム本文を含める。
- SiteMinder の理由およびリダイレクト URL を転送するために使用できるカスタム本文を含める。

デフォルト： 値なし。 この値を省略した場合、SiteMinder は本文なしで Web アプリケーション クライアントに対するレスポンスを転送します。

ContentType

レスポンスが含まれるファイルに存在するデータの MIME 形式を指定します。

デフォルト： 値なし。 この値を省略した場合、テキスト/プレーンの MIME タイプがパラメータに適用されます。

カスタム本文に SiteMinder によって生成されたレスポンスが含まれる場合、データのコンテンツ タイプは以下のいずれかのタイプである必要があります。

- text/*
- application/xml
- application/*+xml

Charset

本文ファイルに存在するデータの文字セットを指定します。

デフォルト： 値なし。この値を省略した場合、パラメータは us–ascii の文字セット タイプを適用します。

Cookie プロバイダと Web アプリケーション クライアント レスポンス

WebAppClientResponse パラメータを設定するときに以下の点を考慮してください。

- Web 2.0 リソースにアクセスする場合、SiteMinder は Cookie プロバイダ上でセッション Cookie を更新しません。
- .html、.jsp、.asp、.cgi などの Web 2.0 以外のリソースにアクセスする場合、SiteMinder は Cookie プロバイダ上のセッションを通常どおりに更新します。

Web アプリケーションへの Web アプリケーション クライアント レスポンスの適用方法

Web アプリケーションと共に Web アプリケーション クライアント レスポンスを適用すると、SiteMinder セキュリティを維持しながら Web アプリケーション クライアントの機能を実装できます。 以下の手順を完全に実行して、Web アプリケーション クライアント レスポンスを適用します。

1. Web アプリケーション クライアント レスポンス（WebAppClientResponse） ACO パラメータを設定します。
2. カスタム レスポンスを設定します。
3. カスタム レスポンスを処理するように Web アプリケーションを設定します。

Web アプリケーション クライアント レスポンスの設定

Web アプリケーション クライアント レスポンスを設定する方法

1. 以下のタスクのいずれかを実行します。
   - 管理 UI で［エージェント設定オブジェクト］（ACO）を開き、WebAppClientResponse のコメントを解除します。
   - ローカル エージェント設定ファイルを開き、WebAppClientResponse のコメントを解除します。
2. 以下の 1 つ以上のデフォルト属性の値を入力します。
   - リソース
   - メソッド
   - ステータス
   - 本体
   - Content–Type
   - Charset

注: 以下の制限について考慮してください。

- この ACO パラメータは、1 つ以上の属性で有効な値を必要とします。
- 追加の属性はすべてオプションです。
- 複数の Web アプリケーションからのリクエストを識別する必要がある場合、単一の ACO パラメータには属性ごとに複数の値を含めることができます。

3. 以下のタスクのいずれかを実行します。
   - 管理 UI で ACO を保存します。
   - ローカル エージェント設定ファイルを保存します。

カスタマイズしたレスポンスの設定

アプリケーションの所有者は、Web エージェント ホスト システム上に存在するファイルの本文内でカスタマイズしたレスポンスを設定します。Web アプリケーション クライアント リクエストが SiteMinder 機能をトリガすると、Web エージェントは Web アプリケーション クライアントに対するレスポンスとして本文を返します。

以下の点を考慮してください。

- ファイルには、アプリケーションの所有者が設計した任意のカスタム本文を含めることができます。
- ファイルはテキストベースにできます。ファイルがテキストベースの場合、SiteMinder は Web アプリケーション クライアントにレスポンスを送信する前に $$Reason$$ と $$URL$$ 用のファイルの本文を解析します。

  レスポンスに SiteMinder によって生成された動作を含める場合：

  - データのコンテンツ MIME タイプは以下のいずれかのタイプである必要があります。
    - text/*
    - application/xml
    - application/*+xml

- 以下のプレースホルダ値が本文に表示される必要があります。

  ```
  SiteminderReason=$$Reason$$
  SiteminderRedirectURL=$$URL$$
  ```

  SiteMinder は、これらの値の本文を解析し、トリガされた SiteMinder 機能とリダイレクト URL を挿入します。 以下のパラメータまたはポリシー レスポンス タイプが機能と URL を定義します。

  - IdleTimeoutURL
  - MaximumTimeoutURL
  - ForceFQHost
  - LogOffRedirectURL
  - ExpiredCookieURL
  - OnAuthAcceptRedirect
  - OnAuthRejectRedirect
  - OnAccessAcceptRedirect
  - OnAccessRejectRedirect
  - Challenge

  例： Web アプリケーション クライアント リクエストがアイドル タイムアウトをトリガすると仮定します。 SiteMinder は、プレースホルダ値を、IdleTimeoutURL と IdleTimeoutURL パラメータの値で指定された URL に置換します。

- ファイルにはバイナリ データを含めることができます。ファイルにバイナリ データが含まれる場合、SiteMinder は解析せずに Web アプリケーション クライアントにファイルの本文を転送します。

カスタム レスポンスを処理するように Web アプリケーションを設定

カスタム レスポンスに SiteMinder の理由およびリダイレクト URL を含める場合は、カスタム レスポンスを処理するように Web アプリケーションを別々に設定します。

Web エージェント インストール ウィザードは、サンプル アプリケーションを *web_agent_home*/samples にインストールします。 特定の環境および状況に合わせてサンプルから推定します。

web_agent_home

Web エージェントのインストール パスを指定します。

セッション猶予期間の変更

通常、Web ページは数多くのリソースで構成され、そのすべてのリソースが Web エージェントによって潜在的に保護されています。 1 つのリクエストに関連付けられている各リソースに対して、1 つのセッション cookie が生成されます。 1 つのユーザ リクエストに対して複数のセッション cookie を生成するオーバーヘッドを取り除くには、以下のパラメータを設定します。

SessionGracePeriod

SiteMinder セッション（SMSESSION） cookie が再生成されない秒数を指定します。 以下の条件が すべて満たされる場合、cookie は再生成されません。

- URL SMSESSION cookie が存在しない。
- 現在の時刻と受け取った SMSESSION cookie の最終アクセス時刻の差が SessionGracePeriod 以下である。
- 現在の時刻と受け取った cookie がアイドルになった時刻の間隔が 2 つの猶予期間を超えている。 たとえば猶予期間が 25 分でアイドル タイムアウトが 60 分である場合、セッションがアイドルになる前に残っている時間が 2 つの猶予期間（50 分）に満たないので、SiteMinder は 10 分後にセッション cookie を再生成します。

デフォルト： 30

セッション猶予期間を変更するには、以下の手順に従います。

1. SessionGracePeriod パラメータの値を変更します。
2. ステップ 1 で SessionGracePeriod パラメータの値を増加した場合は、管理 UI を使用して、すべてのレルムで以下の値の両方が SessionGracePeriod パラメータの値を超えていないことを確認します。
   - セッション タイムアウト値
   - アイドル タイムアウト値

   セッション猶予期間が変更されます。

   注: セッション タイムアウトは、レルムの設定の一部なので、管理 UI を使用して設定します。セッション タイムアウトの設定方法の詳細については、ポリシー サーバのマニュアルを参照してください。

セッション更新期間の変更

以下のパラメータを使用して、Web エージェントが cookie プロバイダにリクエストをリダイレクトして新しい cookie を設定する間隔を指定することができます。

SessionUpdatePeriod

新しい cookie を設定する目的で、Web エージェントが要求を cookie プロバイダにリダイレクトする頻度(秒単位)を指定します。マスタ cookie を更新すると、SiteMinder セッションのアイドル タイムアウトが原因でその cookie が期限切れになる確率を低下させることができます。

デフォルト： 60

セッション更新期間を変更するには、以下の手順に従います。

1. CookieProvider パラメータが定義されていることを確認します。
2. 目的の間隔を反映するように、SessionUpdatePeriod パラメータの秒数を変更します。

   セッション更新期間が変更されます。

検証期間と期限切れになった Cookie URL での悪用からのセッション Cookie の保護

SiteMinder では、以下のアイテムにアクセスできる管理者やその他のユーザによって SiteMinder セッション cookie が侵害される可能性を大幅に減らすことができる時間ベースのセッション cookie パラメータが使用されます。

- Web サーバ ログ
- SiteMinder Web エージェント ログ
- クロスドメイン シングル サインオンの場合にドメイン間に置かれている、侵害される可能性のあるプロキシ サーバ

これらの時間ベースのセッション cookie パラメータは、セッション cookie に「生成日」の概念を付加します。 リダイレクトの結果としてセッション cookie（URL セッション cookie）を受け取るエージェントは、cookie の生成日名と値のペアを探し、この値を設定パラメータ CookieValidationPeriod に設定されている値と比較します。 生成日の値に CookieValidationPeriod パラメータの値を加えた値が現在の時刻に達しない場合、cookie は拒否されます。

悪用からセッション cookie を保護するには、以下のパラメータを設定します。

CookieValidationPeriod

エージェントがセッション cookie 受け取る期間を（秒単位で）指定します。 この期間を過ぎると、セッション cookie は受け取られません。 このフィールドが使用されていないか、ゼロに設定されている場合、アイドル タイムアウトおよび最大セッション タイムアウト値に達すると、セッション cookie は期限切れになります。

デフォルト： 空白.

ExpiredCookieURL

（省略可）セッション cookie の期限切れ後にエージェントがユーザをリダイレクトする先の URL を指定します。 セッションの作成日も CookieValidationPeriod も設定されていない場合、エージェントはこの設定を無視し、cookie を通常どおり処理します（後方互換性）。

セッション Cookie の作成または更新の防止

Microsoft Outlook Web Access など、一部の Web アプリケーションでは、ユーザがアプリケーションをアクティブに使用していない場合でも、HTTP リクエストがバックグラウンドで行われます。 たとえば、ユーザがサーバ上で新しい電子メールをアクティブに確認していない場合でも、Web Access アプリケーションは HTTP リクエストを行います。

ユーザがアイドル状態だったとしても、セッションが期限切れにならないように、これらのリクエストによって SMSESSION cookie が更新されることがあります。 セッションが通常どおり期限切れになるように、これらのバックグラウンド リクエストの際に Web エージェントがセッション cookie を作成または更新できないようにすることができます。

以下のパラメータを設定します。

OverlookSessionForMethods

Web エージェントがこのパラメータ内に列挙されたメソッドに対してすべての HTTP リクエストのリクエスト メソッドを比較するかどうかを指定します。 一致した場合、Web エージェントは SMSESSION cookie の作成も更新も行いません。 さらに、cookie プロバイダ（設定されている場合）はそのリクエストに対して更新されません。

デフォルト： デフォルトなし

OverlookSessionForUrls

Web エージェントが、すべての HTTP リクエストの URLs を、このパラメータに示されている URLs と比較するかどうかを指定します。 一致した場合、Web エージェントは SMSESSION cookie の作成も更新も行いません。 さらに、cookie プロバイダ（設定されている場合）はそのリクエストに対して更新されません。

デフォルト： デフォルトなし

例： /MyDocuments/index.html のような相対 URL を使用します。絶対 URL（http://fqdn.host/MyDocuments/index.html）は使用しません。

注: 前述のパラメータの両方を設定すると、URL の前にメソッドが処理されます。

OverlookSessionAsPattern

有効な場合、Web エージェントは、OverlookSessionForUrls で指定されるディレクトリ下にあるどの URL 用の Cookie も作成しません。

デフォルト： No

値： Yes、No

例： OverlookSessionForUrls で /siteminder を指定し、OverlookSessionAsPattern を［Yes］に指定すると、Cookie はどの /siteminder/* リクエストに対しても生成されません。

メソッドと URI に基づいたセッション Cookie の作成または更新の防止

Microsoft Outlook Web Access など、一部の Web アプリケーションでは、ユーザがアプリケーションをアクティブに使用していない場合でも、HTTP リクエストがバックグラウンドで行われます。 たとえば、ユーザがサーバ上で新しい電子メールをアクティブに確認していない場合でも、Web Access アプリケーションは HTTP リクエストを行います。

ユーザがアイドル状態だったとしても、セッションが期限切れにならないようにこれらのリクエストによって SMSESSION cookie が更新されます。セッションが一般的に期限切れになるように、これらのバックグラウンドリクエストの際に Web エージェントがセッション cookie を作成または更新できないようにすることができます。

メソッドと URI に基づいた作成または更新を防ぐ方法

1. 以下のパラメータをすべて設定します。

   OverlookSessionForMethods

   Web エージェントがこのパラメータ内に列挙されたメソッドに対してすべての HTTP リクエストのリクエスト メソッドを比較するかどうかを指定します。 一致した場合、Web エージェントは SMSESSION cookie の作成も更新も行いません。 さらに、cookie プロバイダ（設定されている場合）はそのリクエストに対して更新されません。

   デフォルト： デフォルトなし

   OverlookSessionForMethodUri

   Web エージェントが、すべての HTTP リクエストの URI を、このパラメータに示されている URI と比較するかどうかを指定します。一致した場合、Web エージェントは SMSESSION cookie の作成も更新も行いません。 さらに、cookie プロバイダ（設定されている場合）はそのリクエストに対して更新されません。

   デフォルト： デフォルトなし

   注: メソッドは URI の前に処理されます。

セキュリティ強化のためにセッション Cookie をセッション ストアに格納する

セッション Cookie はエンド ユーザのクライアント コンピュータに格納されます。 SiteMinder セッション ストアに格納されるセッション Cookie を SiteMinder に作成させることによって、環境のセキュリティを強化することができます。 SiteMinder セッション ストアにセッション Cookie を格納することにより、以下のアイテムへのアクセス権を持つ第三者がクライアント コンピュータからセッション Cookie をコピーし、次にリプレイ攻撃を試行するのを妨げます。

- Web サーバ ログ
- SiteMinder Web エージェント ログ
- ドメイン間に置かれている、侵害される可能性のあるプロキシ サーバ（複数のドメイン間でのシングル サインオンの場合）

次のパラメータを設定することにより、SiteMinder がセッション Cookie を格納する場所を制御できます。

StoreSessioninServer

クライアント コンピュータ上、または SiteMinder セッション ストアにセッション Cookie が格納されるかどうかを指定します。 StoreSessioninServer パラメータの値が Yes の場合は、セッション Cookie が作成され、セッション ストアに格納されます。 Cookie プロバイダおよび Web エージェントは、セッション ストアの Cookie にアクセスします。

Cookie プロバイダおよび Web エージェントは、URL 内のセッション Cookie を、セッション ストアに格納されたセッション Cookie に対応する GUID に置き換えます。

StoreSessioninServer パラメータの値が No の場合、セッション Cookie は URL で直接渡されます。

デフォルト： No

次の手順に従ってください:

1. 環境が以下の条件を満たすことを確認します。
   - SiteMinder 6.0 SP5 QMR1 以上を使用するために、Web エージェントと Cookie プロバイダをアップグレードする。

- Web エージェントおよび Cookie プロバイダ内で DefaultAgentName パラメータの値を使用する。
- ポリシー サーバが有効なセッション ストアで設定されている。

2. Web エージェントおよび Cookie プロバイダで、StoreSessioninServer パラメータの値を Yes に設定します。

セッション Cookie ドメインの検証

以下のパラメータを使用して SiteMinder にセッション cookie のドメインを検証させることにより、不正なユーザがハイジャックし、SiteMinder セッション cookie を再利用しようとするリスクを減らすことができます。

TrackSessionDomain

セッション cookie の中にセッション cookie の対象ドメインを暗号化して格納するように Web エージェントに指示します。 後続のリクエストでセッション cookie が提示されると、Web エージェントは、セッション cookie 内にある対象ドメインを、要求されたリソースのドメインと比較します。ドメインが一致*しない*場合、Web エージェントはリクエストを拒否します。

たとえば、このパラメータの値が yes に設定されているときに、operations.example.com での使用を目的とするセッション cookie が finance.example.com で提示された場合、Web エージェントはその cookie を拒否します。

デフォルト： no

SiteMinder にセッション cookie のドメインを検証させるためには、TrackSessionDomain パラメータの値を yes に設定します。

セッション タイムアウト後のユーザのリダイレクト

ユーザが 管理 UI でレルムを設定すると、セッション タイムアウトが設定されます。ユーザの SiteMinder セッションがタイムアウトしたとき、Web エージェントは以下の処理のいずれかを行います。

- ユーザに認証情報を再要求する
- ユーザを別の URL へリダイレクトする

リダイレクト URL が指定されている場合、ユーザにはその宛先ページが表示されます。 ページが保護されていない場合は、そのページへの直接アクセス権がユーザに付与されます。ページが保護されている場合は、ページへのアクセス権の付与に先立って、ユーザに対して認証情報が要求されます。 リダイレクト URL が指定されていない場合、Web エージェントはセッション タイムアウト後にユーザに認証情報を再要求します。

セッションがタイムアウトしたユーザをカスタマイズされた Web ページの URL にリダイレクトできます。カスタマイズされた Web ページでは、セッションが終了した理由とセッションを再確立する方法が説明されます。 たとえば、「You have been logged out automatically as a security precaution. Please login again to continue.」などのメッセージが表示されるカスタム Web ページを作成できます。

セッションがタイムアウトになった後、ユーザが別ページにリダイレクトされなければ、SiteMinder は再度ユーザに認証を要求します。 再認証が要求されている理由が理解できないため、これによってユーザが混乱する可能性があります。

セッション タイムアウトの後にユーザを別の URL にリダイレクトする方法

1. エージェント設定オブジェクトまたはローカル設定ファイルに以下のパラメータを追加します。

   IdleTimeoutURL

   セッションのアイドル タイムアウトが発生したときに、Web エージェントがユーザをリダイレクトする先の URL を指定します。

   例： http://example.mycompany.com/sessionidletimeoutpage.html

   注: IdleTimeoutURL は、非永続セッションにのみ使用してください。永続セッションに対して設定した場合は無効になります。

   MaxTimeoutURL

セッションの最大タイムアウトが発生したときに、Web エージェントがユーザをリダイレクトする先の URL を指定します。

例： http://example.mycompany.com/maxtimeoutpage.html

デフォルト： デフォルトなし

2. 前のパラメータごとに 1 つずつ URL を入力します。すべてのパラメータに対して同じ URL を使用することも、それぞれ異なる URL を使用することもできます。

   IdleTimeoutURL と MaxTimeoutURL が設定されている場合に、（ポリシー サーバで設定されている）セッションのアイドル タイムアウト値と最大タイムアウト値に同時に達すると、タイムアウトが発生したときにユーザは MaxTimeoutURL に指定された URL にリダイレクトされます。

複数レルム間でタイムアウトを適用する方法

ユーザ セッション タイムアウトは、ユーザが最初にログオンしたレルムによって制御されます。 シングル サインオンによってユーザが新しいレルムに入った場合、新しいレルムに関するタイムアウト値は、引き続き、最初のレルムに初期のログインをした際に確立されたセッションによって制御されます。 別のレルムに対する別のタイムアウト値があり、各レルムにそれぞれのタイムアウト値を使用させる場合は、元のレルムのタイムアウトを無視できます。

既にタイムアウトになったユーザは、他のレルムにログインする際に、認証情報を再度要求されます。 たとえば、Realm1 の Idle Timeout が 15 分であり、Realm2 の Idle Timeout が 30 分である場合、ユーザが Realm1 で 20 分のアイドル時間を過ごした後、Realm2 にログインしようとすると、認証情報を要求されます。

元のレルムのタイムアウトを無視するには、Web エージェントとレルムを以下の手順で説明するように設定します。

1. EnforceRealmTimeouts パラメータの値を yes に設定します。
2. 管理 UI を使用して以下のタスクを実行します。
   a. 元のタイムアウトに代わるレルム（SSO 機能がユーザのアクセスを許可するレルム）ごとに、以下を実行します。
      - 最大タイムアウト値を無視するには、WebAgent-OnAuthAccept-Session-Max-Timeout レスポンス属性を使用して、レスポンスを作成します。
      - アイドル タイムアウト値を無視するには、WebAgent-OnAuthAccept-Session-Idle-Timeout レスポンス属性を使用して、レスポンスを作成します。
   b. 前のレスポンスをそれぞれ OnAuthAccept ルールにバインドします。

   注: 作成するレスポンスの詳細については、「ポリシー サーバ構成ガイド」を参照してください。

複数の有効なセッションが存在する場合、レルム タイムアウトの後に再チャレンジを防ぐ

旧バージョンの SiteMinder は、レルム タイムアウト発生時に自動的にユーザにクレデンシャルを再チャレンジしました。 複数のセッションがポリシー サーバに存在したときさえ、このチャレンジが発生しました。

このバージョンでは、ポリシー サーバにユーザにチャレンジする前にリストですべてのセッションを調べさせるオプションがあります。

以下のパラメータがこのオプションを制御します。

compatRealmtimeouts

レルム タイムアウト発生後にポリシー サーバがユーザにクレデンシャルをチャレンジするかどうかを指定します。 このチャレンジは、ポリシー サーバの最初のセッションが期限切れのために発生します。 ポリシー サーバはリストの他の関連セッションを検査しません。 このパラメータの値が yes である場合、ポリシー サーバはリストの最初のセッションのみを確認します。 ユーザはチャレンジされます。 このパラメータの値が no である場合、ポリシー サーバはユーザにチャレンジする前にリストのすべてのセッションを確認します。

デフォルト： No（レルム タイムアウト時にすべてのセッションが確認されます）

レルム タイムアウト発生時に、リストの最初のセッションのみを調べるには、compatRealmtimeouts パラメータの値を yes に変更します。

クライアント証明書を SiteMinder セッションにリンクする方法（Windows）

IIS 7.x（Windows のみ）および Apache ベースの Web サーバについては、クライアント証明書を SiteMinder セッションとリンクできます。この機能は、以下のアイデンティティの一致を確認します。

- SiteMinder セッションと関連付けられるユーザのアイデンティティ。
- トランザクションで使用されるクライアント証明書のアイデンティティ。

これらのアイテムが一致しない場合、製品はトランザクションをブロックします。

この機能を使用するには、以下のタスクを実行します。

- クライアント証明書を自動的に取得するように、すべての Web サーバを設定します。 特に、ログオン- サーバがポリシー-実行ポイントから離れている場合。

X.509 証明書認証方式（他の認証方式がサポートされていない）を使用します。

以下の図は、クライアント証明書をセッションとリンクする方法を説明しています。

次の手順に従ってください:

1. プラグインを WebAgent.conf ファイルに追加します (P. 145)。
2. エージェント設定パラメータを設定します (P. 146)。

プラグインの追加

プラグインの追加はクライアント証明書をセッションとリンクする最初の手順です。

次の手順に従ってください:

1. エージェントをホストするシステムへログインします。
2. テキスト エディタで以下のファイルを開きます。

   `WebAgent.conf`
3. 次の行を検索します。

   `LoadPlugin="web_agent_home¥bin¥HttpPlugin.dll"`
4. 手順 3 でこの行のすぐ後に行を追加します。
5. 新規行へ以下のテキストを追加します。

   `LoadPlugin="web_agent_home¥bin¥CertSessionLinkerPlugin.dll"`

   注: CertSessionLinkerPlugin が HttpPlugin の後に続く必要があります。
6. ファイルを保存します。
7. Web サーバを再起動します。

   プラグインが追加されます。設定パラメータを追加して、続行します。

プラグインを追加した後に、エージェント設定パラメータを設定します。

次の手順に従ってください:

1. 管理 UI を使用して、目的のエージェント設定オブジェクトを開きます。
2. 以下のパラメータの値を変更します。

   CslCertUniqueAttribute

   それによって一意に識別される証明書の属性をリスト表示します。以下の証明書属性が使用できます。

   - version
   - serialnumber
   - signaturealgorithm
   - issuerdn
   - subjectdn
   - validitystart
   - validityend

   注: このパラメータ内の値の順番は重要ではありません。

   デフォルト： 無効（serialnumber および issuerdn 属性のみ一致）。

   CslMaxCacheEntries

   エージェント キャッシュに含められるエントリの最大数を指定します。

   注: UNIX で動作する Apache ベースのサーバについては、singleprocessmode パラメータの値を「no」に設定することをお勧めします。この設定によって、複数のリクエスト間で共有されるマルチ- プロセッサが作成されます。 Apache ベースのサーバがプレ-フォーク モードで実行される場合、この設定によってパフォーマンスが改善されます。

   デフォルト： 1000
3. 変更内容を保存し、エージェント設定オブジェクトを閉じます。

   証明書はセッションとリンクされます。

クライアント証明書をセッションにリンクする方法（Windows）

IIS 7.x（Windows のみ）および Apache ベースの Web サーバについては、クライアント証明書を SiteMinder セッションとリンクできます。この機能は、以下のアイデンティティの一致を確認します。

- SiteMinder セッションと関連付けられるユーザのアイデンティティ。
- トランザクションで使用されるクライアント証明書のアイデンティティ。

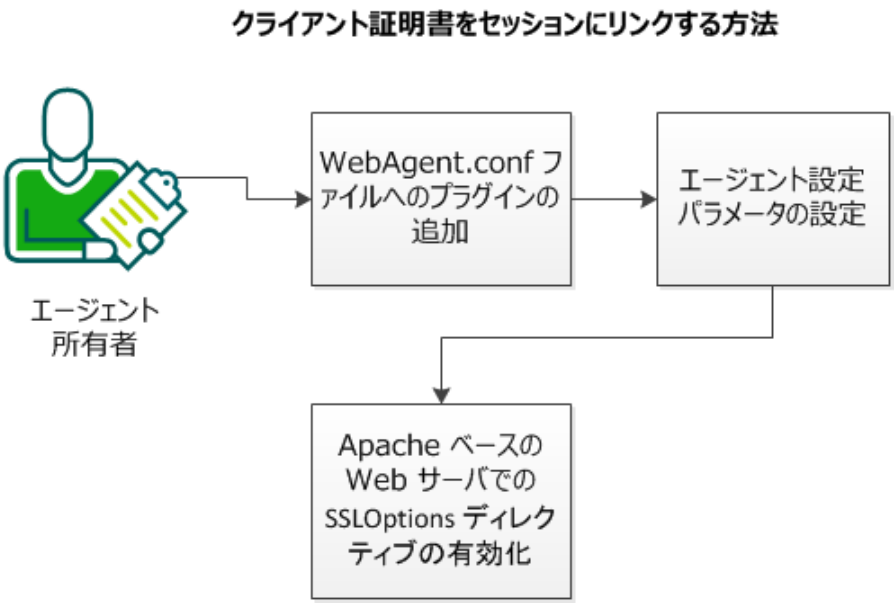
これらのアイテムが一致しない場合、製品はトランザクションをブロックします。

この機能を使用するには、以下のタスクを実行します。

- クライアント証明書を自動的に取得するように、すべての Web サーバを設定します。 特に、ログオン- サーバがポリシー-実行ポイントから離れている場合。

X.509 証明書認証方式（他の認証方式がサポートされていない）を使用します。

以下の図は、クライアント証明書をセッションにリンクする方法を説明しています。



次の手順に従ってください:

1. プラグインを WebAgent.conf ファイルに追加します (P. 148)。
2. エージェント設定パラメータを設定します (P. 149)。
3. Apache ベースの Web サーバで SSLOptions ディレクティブを有効にします。 (P. 151)

プラグインの追加

プラグインの追加は、クライアント証明書をセッションにリンクする最初の手順です。

次の手順に従ってください:

1. エージェントをホストするシステムへログインします。
2. テキストエディタで以下のファイルを開きます。

   `WebAgent.conf`
3. 次の行を検索します。

   `LoadPlugin="web_agent_home/bin/libHttpPlugin.so"`
4. 手順 3 でこの行のすぐ後に行を追加します。
5. 新規行へ以下のテキストを追加します。

   `LoadPlugin="web_agent_home/bin/libCertSessionLinkerPlugin.so"`

   注: libCertSessionLinkerPlugin が libHttpPlugin の後に続く必要があります。
6. ファイルを保存します。
7. Web サーバを再起動します。

   プラグインが追加されます。設定パラメータを追加して、続行します。

プラグインを追加した後に、エージェント設定パラメータを設定します。

次の手順に従ってください:

1. 管理 UI を使用して、目的のエージェント設定オブジェクトを開きます。

   以下のパラメータの値を変更します。

   CslCertUniqueAttribute

   それによって一意に識別される証明書の属性をリスト表示します。以下の証明書属性が使用できます。

   - version
   - serialnumber
   - signaturealgorithm
   - issuerdn
   - subjectdn
   - validitystart
   - validityend

   注: このパラメータ内の値の順番は重要ではありません。

   デフォルト： 無効（serialnumber および issuerdn 属性のみ一致）。

CslMaxCacheEntries

エージェント キャッシュに含められるエントリの最大数を指定します。

注: UNIX で動作する Apache ベースのサーバについては、singleprocessmode パラメータの値を「no」に設定することをお勧めします。この設定によって、複数のリクエスト間で共有されるマルチ- プロセッサが作成されます。 Apache ベースのサーバがプレ-フォーク モードで実行される場合、この設定によってパフォーマンスが改善されます。

デフォルト： 1000

2. 任意の Apache ベースのサーバに対して以下のパラメータが存在する場合は、値が no であることを確認します。

SingleProcessMode

CSL キャッシュが Apache ベースのサーバ（ワーカ モード）上の単一のプロセスでのみ使用可能かどうかを指定します。 キャッシュを単一のプロセスでのみ使用可能にするには、この値を「yes」に設定します。 UNIX 動作環境では、Apache ベースのサーバはプリ-フォーク モードで動作します。このモードでは、個別のプロセスが各リクエストを処理します。CSL キャッシュが複数のリクエスト間で共有されるように、UNIX 動作環境でこの値を「no」に設定することをお勧めします。

デフォルト： No （プリ-フォーク モード、マルチ-プロセス キャッシュ）

オプション： Yes （ワーカ モード、単一プロセス キャッシュ）

3. 変更内容を保存し、エージェント設定オブジェクトを閉じます。

SSLOption ディレクティブを有効にして、続行します。

Apache ベースの Web サーバで SSLOptions ディレクティブを有効にします。

次の手順は、Apache- ベースのサーバの SSLOptions ディレクティブを有効にすることです。 このディレクティブを有効にすると、証明書属性が環境変数として使用可能になります。

Apache ベースのサーバの httpd.conf ファイルに以下のエントリを追加します。

```
SSLOptions +StdEnvVars
```

第 9 章: Web アプリケーションの保護

このセクションには、以下のトピックが含まれています。

Web アプリケーション開発用メカニズム

SiteMinder には、Web アプリケーションを保護するために次の方法が用意されています。

- 認証されたユーザ名をアプリケーションへ渡すための REMOTE_USER 変数 (P. 153)。
- レスポンス属性 (P. 158)。
- HTTP ヘッダ (P. 162)。
- カスタム エラー ページ (P. 179)。

REMOTE_USER 変数

REMOTE_USER 変数は、Web サーバによって認証されたユーザの名前を保持します。 エージェントが Web サーバにインストールされると、SiteMinder では Web サーバのネイティブ認証を置換します。REMOTE_USER 変数は空白です。

ご使用のアプリケーションが REMOTE_USER 変数を使用する場合は、REMOTE_USER 変数が設定されます。

Web サーバが REMOTE_USER 変数を使用しない場合は、HTTP_SM_USER ヘッダがユーザ名をアプリケーションへ渡す代替方法を提供します。

詳細情報

REMOTE_USER 変数を設定するように Web エージェントを設定する (P. 154)

REMOTE_USER 変数を設定するように Web エージェントを設定する

REMOTE_USER 変数を設定するように Web エージェントを設定する

- REMOTE_USER を SiteMinder のログイン ユーザ名の値に設定するには、Web エージェントの SetRemoteUser パラメータを yes に設定します。

  このパラメータのデフォルト値は no で、これは REMOTE_USER 変数を空白のままにするものです。

  注: SiteMinder Web エージェント 5.x QMR 2 以前では、SetRemoteUser パラメータが影響するのは IIS Web サーバのみでした。Apache および Oracle iPlanet エージェントでは、REMOTE_USER は必ず SiteMinder のログイン ユーザ名に設定されます。 5.x QMR 2 より前のエージェントをインストールするユーザ、またはそのようなエージェントからのアップグレードを行うユーザは、REMOTE_USER がデフォルトで有効にならないことに注意してください。

- REMOTE_USER 変数を、ユーザのログイン認証情報ではなく、特定のユーザ アカウントに基づいて設定するには、以下の手順に従います。

  - SetRemoteUser パラメータを yes に設定して、これを有効にします。
  - RemoteUserVar パラメータを設定します。このパラメータは、エージェントに対し、HTTP-WebAgent-Header-Variable レスポンス属性の値に基づいて 変数の値を設定するように指示するものです。このパラメータは、レガシー アプリケーションと統合するために使用します。

    RemoteUserVar パラメータを設定するには、レスポンス変数の名前のみを入力します。 たとえば、「user=ajohnson」のような HTTP-WebAgent-Header-Variable を返すには、RemoteUserVar パラメータを値「user」に設定します。

- ヘッダ変数を OnAuthAccept ルールにバインドします。 既存の HTTP ヘッダ変数レスポンスを使用せずに、新規作成してください。

  注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

- デフォルトに戻して、REMOTE_USER を空白のままにするには、SetRemoteUser パラメータを no に戻します。

注: SetRemoteUser または RemoteUserVar を設定する前に、セキュリティ上の因果関係を必ず考慮してください。

IIS Web サーバおよび REMOTE_USER 変数

IIS Web サーバでは、SiteMinder で REMOTE_USER 変数を使用するために基本認証が必要です。

基本認証が有効になっている場合に、ユーザが <STMNDR > で保護されたリソースを要求すると、エージェントはユーザ名のみで IIS Web サーバの HTTP_Authorization ヘッダを設定しようとします。 パスワードは使用しません。

HTTP_Authorization ヘッダが使用される場合、IIS Web サーバの基本認証メカニズムは他のすべての認証チャレンジに対して優先されます。 そのため、IIS Web サーバは、ユーザが独自の認証要求に応答しようとしていると見なします。

エージェントが ISAPI フィルタ（IIS のクラシック パイプライン モードを使用）として動作する場合、エージェントは以下のタスクを実行します。

- 要求のユーザ コンテキストを設定します。
- REMOTE_USER ヘッダの値を設定します。

SetRemoteUser パラメータの値が yes で、以下のいずれかの設定が使用される場合、エージェントは REMOTE_USER ヘッダのデータを入力します。

- DefaultUsername および DefaultPassword - これら両方のパラメータにより、エージェントが大部分のアクティビティで使用する（特権）プロキシ ユーザ アカウントが制御されます。
- ForceIISProxyUser - 通常の動作を無効にして、エージェントに、IIS Web サーバをプロキシ サーバとして実行するように強制します。
- UseAnonAccess - エージェントに対して、要求のユーザ コンテキストをまったく提供せずに、既存のユーザ コンテキストを未変更のままにするように指示します。
- 認証済みユーザ セキュリティ コンテキストで実行 - エージェントは IIS Web サーバに対して、永続的なセッションに格納された認証情報を使用するように指示します。

SetRemoteUser パラメータおよび UseAnonAccess パラメータを一緒に使用する場合は、注意してください。

以下の表に、これらのパラメータがどのように連携するかを示します。

パラメータの設定	その結果
SetRemoteUser=yes UseAnonAccess=yes	エージェントはユーザ セキュリティ コンテキストを渡さないため、REMOTE_USER 変数は設定できません。 ユーザ セキュリティ コンテキストがない場合、IIS Web サーバは、エージェントが変更した HTTP_Authorization ヘッダの認証情報を強制的に使用します。ただし、これにはユーザ名しか含まれないため、このヘッダは不完全です。
SetRemoteUser=yes UseAnonAccess=no	エージェントは、DefaultUserName、DefaultPassword、または ForceIISProxyUser などのその他のパラメータの設定方法に応じて、一部のタイプのユーザ コンテキストを渡すことができます。 エージェントがセキュリティ コンテキストを IIS Web サーバへ渡す場合、IIS Web サーバはエージェントの認証情報を使用します。IIS Web サーバは不完全な HTTP_Authorization ヘッダを無視します。

Web エージェントでのレスポンス属性の機能

SiteMinder のレスポンス属性は、アプリケーションに対して、ユーザ データの収集方法や、その情報を適用してユーザごとにパーソナライズしたコンテンツを表示する方法を指示します。

SiteMinder は、設定可能なレスポンス属性を用意しています。これらの属性は、アプリケーションにデータを渡し、ユーザの操作をカスタマイズするための手段です。

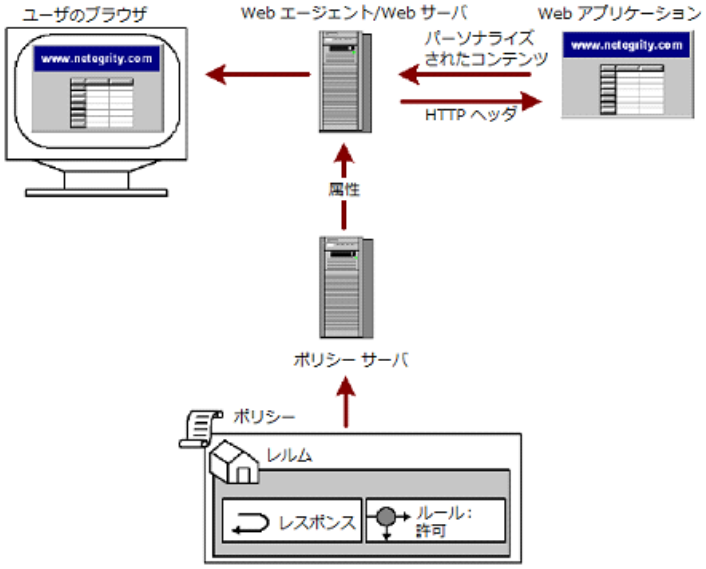
管理 UI を使用してレスポンスを設定し、ポリシー内の特定のルールにそのレスポンスを関連付けます。 リクエストが、設定済みのレスポンスに関連付けられているルールをトリガした場合、ポリシー サーバはそのレスポンス データをエージェントに送信し、エージェントはその情報を解釈し、Web アプリケーションが利用できるようにします。

レスポンスを設定した後、そのレスポンスをエージェントのアクションに関連付けます。 HTTP ヘッダと cookie のレスポンス属性を、GET と POST の各アクションに関連付けることができます。 これらの属性を、認証イベントまたは許可イベントに結び付けることもできます。 ユーザがそれらのイベントのどちらかを受け付けまたは拒否した場合に、ポリシー サーバは 1 つのレスポンスを送信することができます。

注: レスポンス属性を設定する場合、Web サーバがエージェントのレスポンスに使用できる最大バッファ サイズは 32 KB であることに注意してください。 レスポンスについて、バッファ サイズ以外の制限事項はありません。

ヘッダ属性や cookie 属性以外のレスポンス属性を使用できるのは、認証イベントまたは許可イベントが発生した場合のみです。それぞれのイベントでユーザが受け入れられた場合でも、拒否された場合でもかまいません。たとえば、1 つのルールに対応させる 1 つの許可**イベント** アクションを選択し、次に 1 つの **WebAgent-OnReject-Redirect** レスポンス属性を設定することができます。許可プロセスにおいてユーザが SiteMinder によって拒否された場合、エージェントはそのユーザを他のページへリダイレクトして、そのページにそのユーザが拒否された理由を示すメッセージを表示することができます。

以下の図は、レスポンス属性がどのようにポリシー サーバから Web サーバに送信されるかを示したものです。



レスポンス属性のメンテナンスを簡素化するには、イベントの種類ごとに別々のレスポンス属性を定義します。 たとえば、OnAccept イベントに対して 1 つのレスポンス属性を定義し、OnReject イベントに対して別のレスポンス属性を定義します。 レスポンス属性を個別に定義することで、属性値の変更が必要なときの属性検索が容易になります。

フォームの認証要求に関する SM_AGENT_ATTR_USRMSG レスポンスの使用

SM_AGENTAPI_ATTR_USERMSG レスポンスを使用すると、カスタム SiteMinder 認証方式の開発者は、ユーザへの認証要求の一部、または他の目的で、カスタム テキストを自らのクライアント アプリケーションへ返すことができます。

v5 QMR3 およびそれ以降で、Web エージェントはフォームによる認証要求を行う際に、SM_AGENTAPI_ATTR_USERMSG レスポンスから得られたテキストを SMUSRMSG cookie へ変換できるようになりました。

認証要求が完了した後で SMUSRMSG cookie が削除されることを保証するために、FCC は以下のように、POST 要求が成功した後でその cookie を消費（ブラウザから削除）します。

- SiteMinder ネイティブモードでは、エージェントはログインに成功した後、リダイレクトを行って元のターゲット URL に戻している最中に、その cookie を削除します。
- SiteMinder 4.x 互換モードでは、エージェントは FORMCRED cookie を生成した後、リダイレクトを行ってターゲット URL に戻している最中に、その cookie を削除します。

注: SMUSRMSG cookie は一定の時間にわたってユーザのブラウザ内に保存されます。また、安全ではない HTTP 接続を介して伝送される可能性もあります。 その結果、機密データを避ける必要があります。

Web エージェントは、フォームによる認証要求を行う際に、SMUSRMSG cookie の中に配置されたテキストを URL エンコード化します。その結果、それらのテキストは、HTTP 伝送を行う際に安全になりますし、半角スペースや他の有害な文字を除去できます。 FCC は、環境からこのテキストを利用できるようになる前に、カスタム FCC 機能でこのテキストをデコードします。

注: URL エンコードは、テキストが SMUSRMSG cookie に置かれていない限り実装されません。

新しい機能を実装するには、カスタム認証方式の開発者は、カスタム フォームをベースとする認証方式を生成する必要があります。 Sm_AgentApi_Login() の呼び出しが SM_AGENTAPI_CHALLENGE を返した場合、エージェントは、要求を行っているユーザを、 Sm_AgentApi_IsProtected() へのレスポンスとして提供された認証方式 URL へリダイレクトすることにより、そのユーザに認証を要求します。

Web エージェントが、HTML フォーム認証方式テンプレートを使用する認証方式を取り扱う場合、そのエージェントはレスポンス属性 SM_AGENTAPI_ATTR_STATUS_MESSAGE を検索します。この属性が見つかった場合、エージェントは適切な SMUSRMSG cookie を生成し、同時に、認証方式 URL へのリダイレクトを行います。ついで、必要な .FCC ソースファイル内で適切なディレクティブが記述されている場合、FCC はフォームを生成する際に、この cookie を使用できます。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

レスポンス属性のキャッシュ

レスポンス属性をキャッシュに格納するか、またはダイナミック データを含む属性を期限切れにして、強制的にポリシー サーバに問い合わせて情報を更新するように、SiteMinder エージェントに指示することができます。スタティック レスポンス属性を設定すると、ポリシー サーバにより、値のキャッシングが認められます。 スタティック値は不変であるため、再計算の必要はありません。 ユーザ属性、DN 属性、またはアクティブ属性を設定した場合は、その値をキャッシュするか、データが最新であるようにするために一定の間隔で再計算させるか、いずれかを選択することができます。

SiteMinder のデフォルトの HTTP ヘッダ

SiteMinder のデフォルトの HTTP ヘッダは、アプリケーションに対して、ユーザ データの収集方法や、その情報を適用してユーザごとにパーソナライズしたコンテンツを表示する方法を指示します。

Web アプリケーション環境の一部として、SiteMinder エージェントは、デフォルトの HTTP ヘッダを Web サーバに送信します。Web サーバは、そのヘッダ情報を Web アプリケーションが使用できるようにします。 これらのヘッダを使用して関数を組み込んで、Web アプリケーションのコンテンツのパーソナライゼーションを行うことができます。 ヘッダには、ユーザ名やユーザが実行を許可されているアクションのタイプなどの情報を格納することができます。

エージェントは、ヘッダが Web アプリケーションから呼び出されたかどうかにかかわらず、それらのヘッダを送信します（無条件に）。しかし、いくつかのヘッダを無効にして、ヘッダのスペースを空けることもできます。

Web エージェントが使用できる SiteMinder のデフォルトの HTTP ヘッダは、以下のとおりです。

HTTP_SM_AUTHDIRNAME

ポリシー サーバがユーザを認証する対象となるディレクトリの名前を示します。管理者は、管理 UI を使用してこのディレクトリを指定します。

HTTP_SM_AUTHDIRNAMESPACE

ポリシー サーバがユーザを認証する対象となるディレクトリ ネームスペースを指定します。 管理者は、管理 UI を使用してこのネームスペースを指定します。

HTTP_SM_AUTHDIROID

ポリシー サーバ データベースのディレクトリ オブジェクト識別子（OID）を示します。

HTTP_SM_AUTHDIRSERVER

ポリシー サーバがユーザを認証する対象となるディレクトリ サーバを示します。 管理者は、管理 UI を使用してこのディレクトリ サーバを指定します。

HTTP_SM_AUTHREASON

認証が失敗した後または 2 回目の認証要求の後に Web エージェントがユーザに返すコードを示します。

HTTP_SM_AUTHTYPE

ポリシー サーバがユーザの ID の確認に使用する認証方式のタイプを示します。

HTTP_SM_DOMINOCN

Domino LDAP ディレクトリを使用してユーザを認証する場合の、ユーザの Domino 正規名を指定します。

例： HTTP_SM_DOMINOCN="CN=jsmith/O=netegrity"

HTTP_SM_REALM

リソースが存在する SiteMinder のレルムを示します。

HTTP_SM_REALMOID

リソースが存在するレルムを識別するレルム オブジェクトの ID を示します。この ID は、サード パーティ製のアプリケーションでポリシー サーバを呼び出す場合に使用します。

HTTP_SM_SDOMAIN

エージェントのローカル cookie ドメインを示します。

HTTP_SM_SERVERIDENTITYSPEC

ポリシー サーバの識別チケットを示します。このチケットがユーザ ID を追跡します。 Web エージェントは、ユーザに合わせてコンテンツをパーソナライズできるように、このチケットを使用して匿名認証方式で保護されたコンテンツにアクセスします。

HTTP_SM_SERVERSESSIONID

ユーザ セッションを識別する一意の文字列を示します。

HTTP_SM_SERVERSESSIONSPEC

ユーザ セッション情報を含むチケットを示します。この情報をデコードできるのはポリシー サーバのみです。

HTTP_SM_SESSIONDRIFT

Web エージェントがキャッシュ内の情報を使用して、セッションをアクティブにしておくことのできる時間の長さを示します。この時間を過ぎると、ポリシー サーバとのセッションが検証されます。このヘッダを設定する場合、ポリシー サーバ上のセッション サーバを有効にしておくこと、およびセッション検証期間を設定しておくことが必要です。

HTTP_SM_TIMETOEXPIRE

SiteMinder セッションの残り時間を示します。

HTTP_SM_TRANSACTIONID

各ユーザ リクエストに対してエージェントが生成した一意の ID を示します。

HTTP_SM_UNIVERSALID

ポリシー サーバが生成したユニバーサル ユーザ ID を指定します。 この ID は顧客に固有であり、アプリケーションに対してユーザを識別しますが、ユーザ ログインとは異なります。

HTTP_SM_USER

認証されたユーザのログイン名を示します。証明書ベースの認証などで、ユーザがログイン時にユーザ名を入力しなかった場合、この変数は設定されません。

HTTP_SM_USERDN

ポリシー サーバによって判別される、認証済みユーザの識別名を指定します。

匿名認証方式では、このヘッダによって GUID （グローバルな固有識別子）が返されます。

HTTP_SM_USERMSG

認証の試行後にエージェントがユーザに提示するテキストを指定します。 認証方式によっては、認証要求時のテキストや認証に失敗した理由が表示されます。

HTTP ヘッダと cookie 変数

WebAgent-HTTP-Header-Variable および WebAgent-HTTP-Cookie-Variable 属性を使用すると、Web エージェントは、名前/値ペアのスタティック リストまたはダイナミック リストをアプリケーションに渡すことができます。名前/値ペアはリソースを要求するユーザに固有であるため、アプリケーションはユーザが参照するコンテンツをカスタマイズできます。

たとえば、管理者が WebAgent-HTTP-Header-Variable レスポンス属性にユーザのフルネームを格納するように設定するとします。 ユーザが保護対象リソースへのアクセスを許可されると、Web エージェントはユーザのフルネームを Web アプリケーションに渡します。 すると、アプリケーションによってユーザの名前が表示されます。これは、顧客との関係構築に役立ちます。

Web アプリケーション環境で、HTTP-Header-Variable レスポンス属性は HTTP_*attribute_name* 変数として表示されます。ここで、*attribute_name* は、HTTP 変数の名前（たとえば、USERFULLNAME）です。名前の一部としてアンダースコア（_） を使用する必要はありません。アンダースコアは、一部のアプリケーション サーバで問題を引き起こすからです。

注: 属性名の一部に使用されているダッシュ（-）をアンダースコア（_）に変換すること、およびすべてのアルファベットが、サーバによって大文字に変換されることがあります。

ヘッダ変数とエンド ユーザ IP アドレス検証

SiteMinder Web エージェントは、最初のリクエストの後に同じユーザからリクエストを受け取ると、リクエストを送信したユーザの IP アドレスをセッション cookie 内で暗号化されている IP アドレスと比較することによって、後続のリクエストと共に送信されたセッション cookie を検証します。 cookie 内のアドレスは、ユーザの初期の要求中にエージェントによって生成されます。

ファイアウォール、ロード バランサ、キャッシュ デバイス、およびプロキシなど、着信ネットワーク トラフィックを平準化して管理するために使用されるメカニズムは、ユーザの IP アドレスを変更したり、すべての受信要求が単一の IP アドレスまたは少数の IP アドレスのグループから発信されているかのように見せたりすることがあります。 その結果、Web エージェントの IP チェックは無効になります。 Web エージェントは、カスタム HTTP ヘッダおよび安全なプロキシ IP アドレスの設定可能なリストを使用して、このようなネットワーク環境で IP チェックを実行できるようになりました。

以下の表に、新しい IP チェック機能の用語をリストします。

用語	定義
HTTP 要求ヘッダ	HTTP 要求の単一の要素を説明する名前/値ペア。
カスタム IP ヘッダ	中間 HTTP ネットワーク アプリケーションまたはハードウェア デバイスがリクエスタの IP アドレスを格納するために使用する、ユーザ定義の HTTP 要求ヘッダ。
IP チェック	最初の要求後に、要求の REMOTE_ADDR を、SMSESSION cookie に格納されている REMOTE_ADDR 値と比較することによって、Web エージェントが要求の認証性をチェックできる機能。 この機能は、IP 検証とも呼ばれます。
REMOTE_ADDR	Web サーバにリクエストを送る HTTP クライアントの IP アドレスを表す Web サーバ変数。 REMOTE_IP または CLIENT_IP とも呼ばれます。 これは、プロキシ サーバ、NAT ファイアウォール、またはその他のネットワーク サービスやデバイスが、リクエスタとターゲット Web サーバの間にある場合のリクエスタ IP アドレスとは異なります。
要求者	HTTP 要求の発信者。通常、ブラウザを使用しているユーザです。

用語	定義
リクエスタの IP アドレス	オリジナルの HTTP 要求を発信しているユーザの IP アドレス。
シングル サインオン	保護された Web サイトに安全にアクセスするためにユーザに求める認証情報入力を、セッション中に 1 回のみで済むようにする機能。
SMSESSION cookie	Web エージェントがシングル サインオン状態を追跡するために使用する HTTP メカニズム。

カスタム ヘッダによる IP アドレスの検証方法

Web エージェントは、REMOTE_ADDR 変数を使用する代わりにカスタム HTTP ヘッダを使用して、ユーザの IP アドレスを判別できるようになりました。 プロキシまたはその他のデバイスがカスタム クライアント IP ヘッダを設定し、Web エージェントが受信要求でこのヘッダを検索するように設定されている場合は、エージェントは、クライアント IP の情報源としてこのヘッダを使用します。

カスタム ヘッダを設定するほかに、プロキシ IP アドレスのリストをセットアップすることもできます。REMOTE_ADDR がプロキシ リストのアドレスと一致する場合は、Web エージェントは、カスタム ヘッダからユーザの IP アドレスを取得します。一致しない場合は、ユーザの IP アドレスは、REMOTE_ADDR から取得されます。

Web エージェントがリクエスタの IP アドレスを解決すると、アドレスは格納され、要求の処理のために使用されます。 アドレスを解決できない場合は、IP アドレスは unknown に設定されます。

Web エージェントは、クライアント IP アドレスが解決された場所を記録して、必要なデバッグを容易に行えるようにします。

IP アドレス検証の設定

以下のパラメータを使用して IP アドレス チェックを実装できます。

CustomIpHeader

リクエスタの IP アドレスを見つけるためにエージェントが検索する HTTP ヘッダを指定します。このパラメータに値が設定されていない場合は、デフォルトは空の文字列になります。 最大長はないため、値は、有効な HTTP ヘッダ値を含む任意の文字列にできます。

デフォルト： No

例： HTTP_ORIGINAL_IP

ProxyDefinition

カスタム HTTP ヘッダを使用する必要があるプロキシ（キャッシュデバイスなど）の IP アドレスを指定します。 このカスタム ヘッダは、エージェントがリクエスタの IP アドレスを解決するのに役立ちます。

デフォルト： デフォルトなし

制限： 文字列には IP アドレスが含まれている必要があります。サーバ名または完全修飾 DNS ホスト名は*使用しないでください*。

RequireClientIP

エージェントがクライアントの IP アドレスを検証するかどうかを指定します。 この値を yes に設定すると、エージェントはブラウザの Cookie 内の IP アドレスがクライアントの IP アドレスに一致することを検証します。 アドレスが一致*しない*場合、403 エラー メッセージがユーザのブラウザに表示されます。 Cookie に IP アドレスが含まれていない場合、ユーザは認証情報を求められます。

デフォルト： No （クライアントの IP アドレスは検証されません）

注: これらの設定は、TransientIPCheck および PersistentIPCheck パラメータとは無関係です。

以前のリリースの Web エージェントによる IP アドレス検証

6.x QMR 2 または 3 以前の Web エージェントの環境では、IP チェックが設定されている場合、シングル サインオンが影響を受ける可能性があります。

v6.x QMR 2 および 5.x QMR 7 より前の Web エージェントは、リクエスタ IP アドレスを解決できないため、これらの Web エージェントによって作成される SMSESSION cookie は、6.x QMR 2 または 3 の Web エージェントによって破棄されることがあります。 これには、SDK を使用して SMSESSION cookie を生成するカスタム エージェント、アプリケーション サーバ エージェント、および SMSESSION cookie を使用するシングル サインオン環境内のその他のすべての SiteMinder エージェントが含まれます。

逆に言えば、6.x QMR 2 および 3 の Web エージェントは、リクエスタの IP アドレスを解決できるため、このアドレスは古いエージェントによって解決されたアドレスとは異なります。

HTTP ヘッダの保存

新しいヘッダが生成されたときに、既存の HTTP ヘッダを置き換えずに保存するように、Web エージェントを設定することができます。 この機能は、名前は同じであっても値が異なる複数の SiteMinder のレスポンスを生成し、ヘッダへの格納が必要なアプリケーションにおいて有効です。 同じ HTTP ヘッダのインスタンスが複数存在する場合、Web サーバはすべての適切なヘッダ値をカンマで区切って 1 つのヘッダを生成して処理します。

デフォルトでは、Web エージェントは、誤ったヘッダ値を使用するアプリケーションへの予防措置としてヘッダを保存しません。 Oracle iPlanet、Domino および Apache Web エージェントで HTTP ヘッダを保存するには、PreserveHeaders パラメータを yes に設定します。デフォルト値は no です。

HTTP ヘッダ リソースのキャッシュ方法の制御

以下のパラメータの設定により、Web エージェントでキャッシュ関連のリクエスト ヘッダを処理する方法を制御できます。

AllowCacheHeaders

Web エージェントで、保護されたリソースに対するリクエストを Web サーバに渡す*前に*、リクエストから以下のキャッシュ関連 HTTP ヘッダを削除するかどうかを指定します。

- if-modified-since
- if-none-match

この設定は、ブラウザがキャッシュされたページを使用するかどうかに影響を及ぼします。 この設定は、自動許可リソース（IgnoreExt パラメータの値を含む）には影響*しません*。 Web サーバとブラウザの設定は、自動許可リソースがキャッシュされるかどうかを決定します。

このパラメータには、以下の値を設定できます。

- Yes -- エージェントはキャッシュ関連の HTTP ヘッダを削除*しません*。セッションを検証するため、SMSESSION Cookie が引き続き追跡されます。セッションが期限切れになると、Web エージェントは更新された SMSESSION Cookie を 304「変更なし」のレスポンスで送信します。このレスポンスは、キャッシュに格納される未変更のリソースに適用されます。 if-modified-since HTTP ヘッダに示される時間により、この動作がいつ発生するかが決まります。

重要: このパラメータを yes に設定すると、適切なキャッシュ制御ヘッダがないパーソナライズされたアプリケーションのページがキャッシュされる場合があります。 これは、予期しない動作を引き起こし、ブラウザが機密データをディスクに保存することを可能にします。

- No -- エージェントは、*保護されている*リソースのリクエストからのみ、キャッシュ関連の HTTP ヘッダを削除します。 Web サーバは、リクエストを無条件として取り扱います。 キャッシュのコンテンツは検証されません。
- None -- Web エージェントは、保護されているリソース*および*保護されていないリソースについて、キャッシュ関連のヘッダをすべて削除します。

終了したセッションの場合、ブラウザはキャッシュされたコンテンツを使用 *しません*。AllowCacheHeaders パラメータの値は無視されます。

このパラメータの設定は以下のパラメータに影響します。

- LogOffUri -- LogOffUri パラメータを使用する場合は、AllowCacheHeaders パラメータの値を no に設定します。設定しない場合は、これらのセッションが正しく終了しません。キャッシュされたログアウト ページは、ユーザに表示される場合があります。

デフォルト： いいえ

制限： Yes、No、None

保護されているまたはされていないリソースからキャッシュ関連ヘッダをすべて削除するには、AllowCacheHeaders パラメータの値を none に設定します。

注: HTTP 1.1 キャッシュの仕組みの詳細については、RFC 2616 でセクション 13 「Caching in HTTP」を参照してください。

HTTP ヘッダのエンコード仕様の設定

HTTPHeaderEncodingSpec の設定は、すべての HTTP ヘッダの値、およびすべてのカスタム HTTP-COOKIE レスポンスそれぞれのエンコードに影響を及ぼします。

このパラメータを使用して、Web アプリケーションをサポートし、ローカライズ済みのテキストを特定のエンコードで表示することを期待できます。 Cookie は、ブラウザとポータルの間で HTTP プロトコルを介してやり取りされるので、HTTP トラフィックが無効と見なす文字が、選択したエンコードによって Cookie の中に書き込まれる場合は、RFC-2047 の HTTPWrapSpec を使用します。

たとえば、RFC-2047 による追加のエンコードを実施しない場合、一部の Shift-JIS 文字は望ましくない結果を招くことがあります。

漢字文字の場合は、SHIFT-JIS のスーパーセットである SECP932 を使用できます。 ほとんどの漢字エンコードおよびデコードに SHIFT-JIS を使用できますが、CP932 ではそれより多くの文字セットがカバーされます。

HTTPWrapSpec を使用する場合、データは最初に HTTPHeaderEncodingSpec に従ってエンコードされ、ついで RFC-2047 の仕様に従って追加エンコードされます。

このパラメータの構文は、次のとおりです。

encoding_spec, wrapping_spec

encoding_spec は、UTF-8、Shift-JIS、EUC-J、または ISO-2022 JP のいずれかのエンコード タイプを表すテキスト文字列です。 エージェントに使用させたいエンコードタイプを指定します。

wrapping_spec は、ラッピング仕様ですが、RFC-2047 にする必要があります。 この変数はオプションですが、ラッピング仕様を記述することを強くお勧めします。ここで選択するエンコード タイプは、HTTP プロトコルと互換性のないバイト コードを生成する可能性があるからです。

2 バイトのエンコード済みデータを含むカスタム HTTP cookie レスポンスを使用している場合、このことが特に当てはまります。たとえば、RFC-2047 で追加エンコードしない場合、一部の Shift-JIS 文字が適切に表示されません。 また、ラッピングは、受信アプリケーションがエンコードされたテキストをより適切に解釈できるように、アプリケーションにエンコードのタイプと性質を伝えます。 たとえば、このパラメータを Shift-JIS,RFC-2047 に設定します。

RFC-2047 を使用する場合、エージェントは最初に、選択されたエンコード仕様に基づいてデータをエンコードし、ついで RFC-2047 仕様に従ってそのデータを追加エンコードします。

注: **HTTPHeaderEncodingSpec** の設定を空白のままにした場合、デフォルトは UTF-8 であり、ラッピングは行われません。

重要: 以下の場合に、エージェントがインストールされているプロキシ コンピュータで HTTPHeaderEncodingSpec ACO パラメータに以下の値を設定します。

```
UTF-8,RFC-2047
```

- SiteMinder エージェントで 管理 UI を保護する場合。
- 管理者の DN 値に英語以外の文字がある場合。

RFC 2047 への準拠の無効化

デフォルトでは、Web エージェントは RFC 2047 に準拠しています。 ただし、ConformToRFC2047 パラメータを no に設定することで、この準拠を無効にできます。

このパラメータが存在しないか、yes に設定されている場合は、Web エージェントは RFC 2047 に準拠しています。

ヘッダでの小文字 HTTP の使用（Oracle iPlanet、Apache、Domino Web サーバ）

大文字と小文字を区別するサーバ アプリケーションが存在している場合、エージェントの HTTP ヘッダで大文字または小文字のいずれを使用するかを指定できます。 Web エージェントのデフォルトは、小文字のヘッダです。

たとえば、Oracle iPlanet Web サーバの場合、http_sm_user のように、デフォルトで HTTP ヘッダ変数は小文字になります。

注: IIS Web エージェントは、この機能は利用できません。IIS は、すべてのヘッダを強制的に大文字にするためです。

小文字のヘッダを使用するには、LowerCaseHTTP パラメータを yes に設定します。 大文字のヘッダ変数が必要な場合、LowerCaseHTTP を no に設定します。

詳細情報:

Oracle iPlanet Web サーバ ログのトランザクション ID の記録 (P. 377)

HTTP ヘッダのレガシー変数の有効化

以下のパラメータを使用して、Web エージェントが HTTP ヘッダに使用する命名規則を指定できます。

LegacyVariables

Web エージェントが HTTP ヘッダ名でアンダースコアを使用するかどうかを指定します。一部の Web サーバ（Sun Java System など）では、HTTP ヘッダでアンダースコア文字を使用すると、一部のアプリケーションで問題が発生します。

このパラメータを no に設定すると、以下の例に示されるように、HTTP ヘッダでアンダースコアは使用されません。

SM*HeaderName*

このパラメータを yes に設定すると、以下の例に示されるように、HTTP ヘッダでアンダースコアが使用されます。

SM_*HeaderName*

デフォルト：（従来のエージェント） Yes

デフォルト：（フレームワーク エージェント） No

レガシー変数を有効にし、Web エージェントに HTTP ヘッダ名でアンダースコアを使用させるには、LegacyVariables パラメータの設定値を yes に設定します。

注: Apache 2.4.x の Web サーバの場合、LegacyVariables パラメータを No に設定して、SMUSER や SMUSERDN などの SiteMinder のデフォルトのヘッダを表示します。

デフォルトの HTTP ヘッダ変数の無効化

システムプラットフォームの多くは、HTTP ヘッダの限界値が 4096 バイトになっています。 この限界値を超えないようにしてカスタム レスポンス変数用のスペースを確保する場合、SiteMinder の一部のデフォルト HTTP ヘッダ変数を無効にすることができます。

デフォルトの変数は以下のカテゴリにグループ化されています。

注: 個々の変数を無効にすることはできません。 いくつかの変数のカテゴリのみ無効にできます。

- 認証ソース変数
  - SM_AUTHDIRNAME
  - SM_AUTHDIRSERVER
  - SM_AUTHDIRNAMESPACE
  - SM_AUTHDIROID
- ユーザ セッション変数
  - SM_SERVERSESSIONID
  - SM_SERVERSESSIONSPEC
  - SM_SERVERIDENTITYSPEC
  - SM_SESSIONDRIFT
  - SM_TIMETOEXPIRE
- ユーザ名変数
  - SM_USER
  - SM_USERDN
  - SM_DOMINOCN

HTTP ヘッダ変数はデフォルトで使用されていますが、それらを無効にするには、以下のいずれかのタスクを実行します。

- 認証ソース変数を無効にするには、DisableAuthSrcVars パラメータの値を yes に設定します。
- ユーザ セッション変数を無効にするには、DisableSessionVars パラメータの値を yes に設定します。

  デフォルト： No

- ユーザ名変数を無効にするには、DisableUserNameVars パラメータの値を yes に設定します。

  注: ユーザが Identity Manager、またはこのカテゴリの変数を使用する可能性のあるアプリケーションを使用している場合は、このパラメータの値を no （有効）に設定するようにしてください。

カスタム エラー処理の指定

カスタムエラー処理を使用すると、エラー情報を各自のアプリケーションと関連付けることができます。 アプリケーションをユーザに合わせてカスタマイズするには、HTTP 500、HTTP 401、および HTTP 403 のエラー ページに表示される HTML テキストを変更するか、HTTP 401 エラーを除き、カスタム エラー ページまたはアプリケーションを示す URL にユーザをリダイレクトします。

カスタム エラー処理で設定できるエラーには、以下の種類があります。

- サーバ エラー - エージェントは、HTTP 500 Web サーバ エラーが原因で表示されるエラー ページに、ServerErrorFile を使用します。 これらのエラー コードは、カスタム エラー ページに対して渡されます。次のものが該当します。
  - Web エージェントが、必要な HTTP ヘッダの値を読み込むことができないために発生するエラー
  - 高度な認証用 cookie が解析されない、もしくはエラーのステータスを含めることができない場合のエラー
  - Web エージェントとポリシー サーバの接続エラー
- アクセス拒否エラー - エージェントは、以下のパラメータで指定されたファイルを使用します。

  Custom401ErrorFile

  401（権限不足）のブラウザ エラーが発生した場合に表示する、カスタマイズされた HTML ページを指定します。 リソースにアクセスするための十分な権限がそのユーザにない場合、そのようなエラーが発生します。

  注: 一部の Web サーバは、ユーザが選択したカスタム テキストに独自のテキストを追加します。 そのため、それらのサーバ用のレスポンス ページはカスタマイズすることはできません。

  デフォルト： デフォルトなし（空白）
- cookie 要求エラー - RequireCookies パラメータが設定済みの場合、Web エージェントは基本認証を実施する間に cookie を設定します。基本認証情報と一緒に、ブラウザからこの cookie が返されなかった場合は、ReqCookieErrorFile パラメータによって指定されたエラー ページが返されます。エージェントはそのユーザがその Web サーバにアクセスすることを拒否します。

- クロスサイト スクリプティング エラー - エージェントは、HTTP 403 クロスサイト スクリプティング エラーが原因で表示されるエラー ページに、CSSErrorFile パラメータの中で指定されたファイルを使用します。クロスサイト スクリプティングは、Web サイトのセキュリティを危うくします。

これらの HTML ファイルまたはアプリケーションを作成したら、Web エージェントに対してカスタム エラー ページまたは URL を指示します。

注: Apache サーバがプロキシ サーバまたはリバース プロキシ サーバとして使用されている場合、Apache エージェントは、カスタム SiteMinder エラー ページではなく、Apache HTTP 標準の 500 エラー ページおよび 403 エラー ページを返します。

エラー処理をセットアップする方法

ユーザに対してアプリケーションがエラー メッセージを表示する方法をカスタマイズするには、次のいずれかのタスクを実行します。

- 以下の HTTP エラーについて、ブラウザが表示する HTML テキストを追加します。
  - 500
  - 401
  - 403
- カスタム エラー ページまたはアプリケーションを指す URL にユーザをリダイレクトします。

HTTP 500 および 403 エラーのみ： ユーザをある URL にリダイレクトするようにエージェントを設定すると、エージェントはエラー コードをその URL に付加します。 以下は、追加された URL の例です。

```
?SMError=error_code,
```

標準的な HTML エラー テキストを追加する場合、以下のタグ間の HTML コードのみを指定できます。

```
<body>
</body>
```

カスタム エラー ページまたは URL に移動するようにエージェントを設定するには、以下のタスクの*いずれか*を実行します。

- テキスト ファイルが存在するパスを指定します。

それぞれのエージェント設定パラメータの値に URL を入力します。

エラーおよび他のイベント、そしてそれぞれのエージェント設定パラメータを次の表にリストします。

カスタム応答を設定するエラーのタイプ	設定パラメータの値
サーバ エラー	ServerErrorFile
アクセス拒否エラー	Custom401ErrorFile
Cookie エラー	ReqCookieErrorFile
CSS 文字エラー	CSSErrorFile

エラーファイルはアプリケーション内のどこにあってもかまいません。

重要: 設定するすべての URL は非保護のカスタム エラー ページのままにしてください。

注: アプリケーションの URL で HTML タグが必要な場合は、タグ内の文字をエンコードします。 HTML 文字のエンコードの詳細については、http://www.cert.org/tech_tips/ を参照してください。

次の例は、エラーファイルの中でのファイルのパスと URL を示しています。この例の中で示す構文は、ローカルの設定ファイルに関するものです。また、エージェント設定オブジェクトでこれらのパラメータを設定できます。

ファイルのパス

```
CSSErrorFile="C:¥error¥error.txt"
ReqCookieErrorFile="C:¥custompages¥error.txt"
ServerErrorFile="C:¥error¥error.txt"
Custom401ErrorFile="C:¥error¥accessdenied.txt"
```

URL：

```
CSSErrorFile="http://www.mycompany.com/error.jsp"
ReqCookieErrorFile="http://www.myorg.com/error.asp"
ServerErrorFile="http://www.mycompany.com/error.jsp"
```

詳細情報

カスタム エラー処理の指定 (P. 179)

カスタム 401 ページに関する注意

- Custom401errorfile パラメータを URL に設定しないでください。
- Custom401errorfile に対応する何らかの値（使用可能かどうかにかかわりなく）が存在している場合、エージェントは 60 秒ごとに、そのファイルが変更されたかどうかを調べます。 しかし、このレスポンスは、性質上、スタティックであることが意図されています。 たとえば、「*user_name* denied」というタイプの動的なメッセージを挿入することはできません。

  Custom401errorfile の値が存在する場合、その値の有効性にかかわりなく再チェックがトリガされるので、エージェントを再起動することなく、エラーを訂正できます。 その訂正結果は、次のチェックの際に取り出されます。
- カスタマイズ済みのメッセージ ファイルのテキストが、他のエラーによって公開されることはありません。 そのファイルのパス名は、起動時、およびエラー発生時にログに記録されます。
- カスタマイズの範囲は、Web サーバによって制限されることがあります。一部の Web サーバは、レスポンスに対して独自のテキストを追加することがあるからです。
- カスタマイズ済みテキスト ファイルのサイズは、システムのファイル サイズの上限のみによって制限を受けます。

第 10 章: 仮想サーバの設定

このセクションには、以下のトピックが含まれています。

仮想サーバ サポートをセット アップする方法

仮想サーバは、物理サーバに設定する論理エンティティです。 論理エンティティは 1 つの独立したサーバとして機能します。仮想サーバを設定すれば、1 つの物理サーバ上で複数の Web サイトをホスティングできます。たとえば、仮想サーバを使って特定のサーバ上に www.mysite.com と www.yoursite.com の両方のサイトをホスティングするようにセットアップすることができます。

仮想サーバには以下の各項目を割り当てることができます。

- 一意の IP アドレス
- 物理サーバと共有する IP アドレス
- 別の仮想サーバと共有する IP アドレス

1 つの Web サーバ インスタンスに設定できる Web エージェントは 1 つのみですが、エージェント ID を設定してすべての仮想サーバを保護できます。 あるユーザが www.mysite.com からサーバにアクセスし、別のユーザが www.yoursite.com からサーバにアクセスする場合、それぞれのサーバは個々のエージェント ID によって保護されます。 仮想サーバごとにエージェント ID を作成すると、サイトごとに固有のレルムとルールを定義できるという利点があります。

Web エージェントに対して定義した設定は、その Web サーバ インスタンスに対して定義したすべての仮想サーバに適用されますが、要求の処理は仮想サーバが互いに独立して行い、ポリシー サーバでは、それぞの仮想サーバ要求を別々に扱います。 仮想サーバおよびその設定方法の詳細については、使用する Web サーバのマニュアルを参照してください。

仮想サーバのサポートを設定するには、以下の*いずれか*のタスクを実行します。

- 各仮想サーバのエージェント ID を定義および追加し、AgentName パラメータの値を指定して、仮想サーバの IP アドレスまたはホスト ヘッダ名に割り当てます。
- 固有のものとして識別されることを必要とする仮想サーバのみに対して、エージェント ID を定義します。
- デフォルトのエージェント名を設定します。

注: Oracle iPlanet Web サーバの複数のインスタンスを使用している場合（HTTP 通信用のサーバと HTTPS 通信用のサーバなど）、2 つの WebAgent.conf ファイルが存在します。 各ファイルが複数のエージェント ID を保持することができます。 （Oracle iPlanet という名前は、以前は Sun ONE および iPlanet と呼ばれていた Web サーバです）。

仮想サーバの Web エージェント ID の割り当て

各仮想サーバに対して、追加の Web エージェントが実際に *「定義」* されているわけではありません。代わりに、Web エージェント ID の *「割り当て」* が行われています。 独特のアクセス要件が存在する仮想サーバを保護する場合、または個別のレルムを保護する場合は、各サーバに固有のエージェント ID を割り当て、他のすべての仮想サーバに対しては、デフォルトのエージェント名を使用します。 このオプションには、SiteMinder のインストールを短時間で設定できるだけでなく、別個に保護する必要があるレルムをホストする仮想サーバをこれまでどおり保護できるという利点があります。

AgentName パラメータとそれに関連付けられた IP アドレスによって、Web サーバ インターフェースと、ポリシー ストア内で定義済みのエージェント名の間のマッピングが実現されます。 Web エージェントは、適用対象となるルールとポリシーの正しいセットを取得するために、適切なエージェント名のコンテキスト内でエージェント API 呼び出しを順に実行する必要があります。 ポリシー ストアへのマッピングでエージェント名または IP アドレスが割り当てられない場合、Web エージェントは、仮想サーバにのみ DefaultAgentName パラメータの値を使用します。

固有のエージェント ID を使って仮想サーバを保護するには、AgentName パラメータを使用して、仮想サーバごとに 1 つの Web エージェントを追加します。 仮想サーバごとに異なる Web エージェントを追加することにより、各仮想サーバに固有のレルムとルールを定義することができます。

Web エージェント ID を割り当てるには、以下の手順に従います。

1. エージェント名と IP アドレスを、カンマで区切って入力します。
2. 仮想サーバが同じ IP アドレスを共有し、異なるポートを使用する場合は、IP アドレスに関連付けられたポート番号（たとえば、112.12.12.1:8080）を指定します。 デフォルトのポートを使用している場合、ポート番号は必要ありません。
3. 複数のエージェントを追加するには、以下の例のように、個々の行にそれぞれ入力します。

```
agentname="agent1,123.123.12.12:8080"
agentname="agent2,123.123.12.12:8081"
agentname="agent3,123.123.12.13"
```

4. エージェント ID を追加する場合は、管理 UI で同じ設定を使用してエージェント ID を定義する必要があります。エージェント ID が、エージェント設定の定義と完全に一致するように、管理 UI で定義されていることを確認してください。

AgentName パラメータに入力がない場合、SiteMinder は、仮想サーバにのみ DefaultAgentName の値を使用します。

注: DefaultAgentName を変更する場合は、エージェントに対する定義と完全に一致するように、管理 UI で定義されていることを確認してください。

Web エージェントで無視する仮想サーバの指定

使用中のサイト内にある 1 台の Web サーバが複数の仮想サーバをサポートしている場合、それらの仮想サーバ上には、Web エージェントで保護したくないリソースが存在している可能性があります。 Web サーバ コンテンツのうち、どの部分を保護する必要があるのか Web エージェントが簡単に識別できるように、以下のパラメータを使用します。

IgnoreHost

Web エージェントで無視するあらゆる仮想サーバの完全修飾ドメイン名を指定します。 そのような仮想サーバ上にあるリソースは自動許可され、どのクライアントが要求を行ったかにかかわらず、Web エージェントは常にそれらのリソースへのアクセスを許可します。 許可は、ポリシーではなく Web エージェントの設定に基づいて決定されます。

IgnoreExt や IgnoreURL の各設定など、自動許可に関する他の項目より先に、無視されるホストに関する上記のリストが最初にチェックされます。 したがって、無視されるホスト上にあるリソースに関しては、ダブルドット ルールがポリシー サーバに対する許可の呼び出しをトリガすることはありません。しかし、拡張子に関するルールがそのようなリソースを無視することはありません。

IgnoreHost パラメータに対する URL エントリのホスト部分は、Web エージェントが読み取る、要求されたリソースのホスト ヘッダと完全に一致する必要があります。

注: この値では、大文字と小文字が区別されます。

URL で特定のポートを使用する場合、ポートを指定する必要があります。

一元管理されたエージェントでは、いくつかのサーバを表わすためにエージェント設定オブジェクトで複数値パラメータを使用します。ローカル設定ファイルで設定されたエージェントでは、ファイル内の個別の行に各ホストを列挙します。

例： （指定したポートと共に表示される URL）

IgnoreHost="myserver.example.org:8080"

例： （ローカル設定ファイル）

IgnoreHost="my.host.com"

IgnoreHost="your.host.com"

デフォルト： デフォルトなし

Web エージェントで無視する仮想サーバを指定するには、次のいずれかのタスクを行います。

- 中央設定では、無視するサーバをエージェント設定オブジェクトに追加します。 サーバが複数ある場合は、パラメータに複数値の設定を使用します。
- ローカル設定では、ローカル設定ファイルでサーバごとに個別の行を追加します。

指定された URL を使用するリソースは、Web エージェントによって無視され、それらのリソースへのアクセスが自動的に付与されます。

詳細情報

複雑な URI の処理 (P. 119)

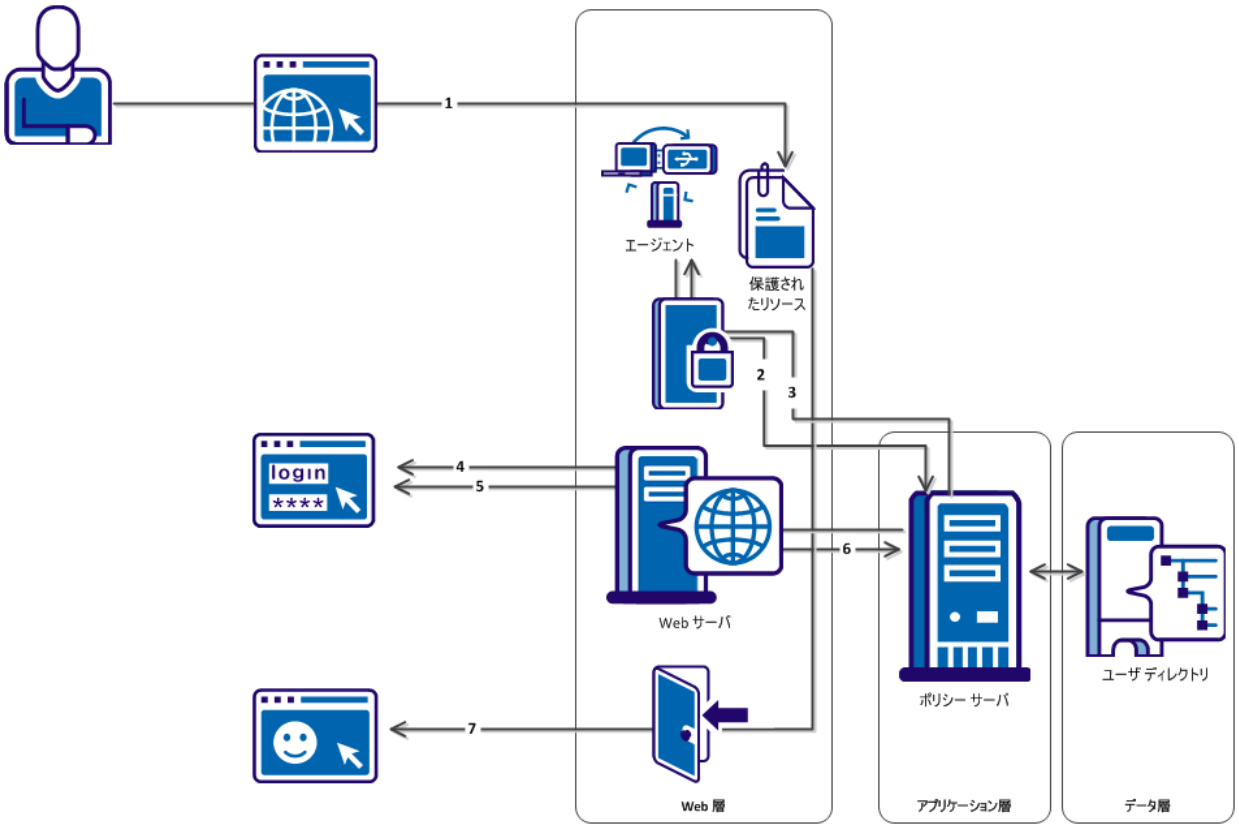
第 11 章: フォーム認証

このセクションには、以下のトピックが含まれています。

認証情報コレクタが要求を処理する方法

以下の図では、フォーム認証情報コレクタ（FCC）がどのように保護リソースへの要求を処理するかについて説明しています。

注: Cookie プロバイダは、シングル サインオンについて別のプロセスを使用します。



前の図で示したプロセスは、以下の手順について説明しています。

1. ユーザがリソースへのアクセスを要求します。
2. Web エージェントはポリシー サーバに確認して、リソースが保護されているかどうかを判断します。
3. ポリシー サーバは、認証情報コレクタがリソースを保護していることをエージェントに伝え、使用している認証情報コレクタのタイプを指定します。
4. エージェントはクエリ データ、ターゲット リソース、および暗号化されたエージェント名を認証情報コレクタの URL に追加します。その後、エージェントはユーザを適切な認証情報コレクタにリダイレクトします。
5. 認証情報コレクタのタイプに応じて、次の*いずれか*のアクションが発生します。
   - FCC はフォームを表示し、次に、ユーザの認証情報を収集します。
   - NTC がユーザの NT 認証情報を収集します。
   - SCC がユーザの認証情報を収集します。
   - 証明書が利用可能でない場合、SFCC はフォームを表示します。SFCC はユーザ認証情報を収集します。
6. 認証情報コレクタは、ユーザをポリシー サーバに直接ログインさせます。 ポリシー サーバは 1 つのセッションを作成します。
7. エージェントはセッションを検証し、ユーザにそのリソースへのアクセス権を付与します。

注: SSL 認証方式の詳細については、ポリシー サーバのマニュアルを参照してください。

認証情報コレクタの MIME タイプ

各認証情報コレクタに対して、1 つの MIME タイプが関連付けられています。その MIME タイプは、ユーザがリソースへのリクエストを行ったときに、どの認証情報コレクタが認証を要求するかを表しています。 次の表は、各タイプを示しています。

認証情報コレクタ	MIME タイプ
フォーム認証情報コレクタ	.fcc
SSL 認証情報コレクタ	.scc
Cookie プロバイダ	.ccc
NTLM 認証情報コレクタ	.ntc
SSL フォーム認証情報コレクタ	.sfcc
Kerberos 認証情報コレクタ	.kcc

認証情報コレクタを使用する認証方式を設定する場合、または複数の cookie ドメイン間でシングル サインオンをセットアップする場合は、その認証方式またはそのシングル サインオンの設定によって参照されるファイル拡張子として、該当の MIME タイプが使用されます。たとえば、次のようになります。

- 複数の cookie ドメイン間でシングル サインオンを設定する場合、次のような 1 つの URL を入力して、cookie プロバイダを識別します。

  http://myserver.company.com:80/siteminderagent/SmMakeCookie.ccc

  SmMakeCookie.ccc は、デフォルトの cookie プロバイダの名前です。この名前を使用すること、または独自の名前を作成することができます。ただし、シングル サインオンを確立するには、拡張子として .ccc を使用する必要があります。

- Windows 認証の場合、この方式を有効にするデフォルトのターゲット ファイルは、以下のとおりです。

  /siteminderagent/ntlm/creds.ntc

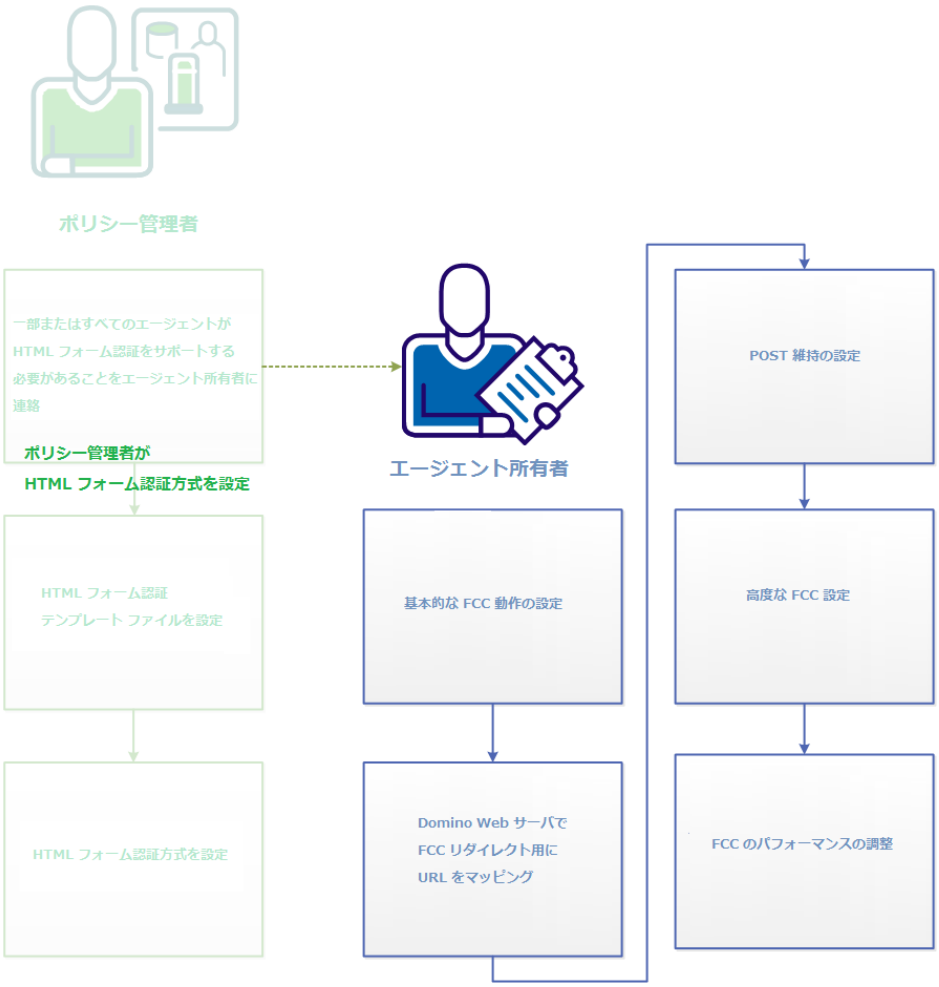
  ここでも、正しい MIME タイプを拡張子としているファイルを使用する必要があります。

エージェントのインストール先である Web サーバ上に実際のファイルが存在していることを必要とする認証情報コレクタは、FCC と SFCC のみです。これらのコレクタは、フォームベースの認証方式を前提としています。ユーザに対して提示される HTML フォームを定義する上で、.fcc と .sfcc の各テンプレートが必要です。

HTML フォーム認証をサポートする SiteMinder エージェントを設定する方法

HTML フォーム認証を使用してユーザ ID を検証するように SiteMinder を設定するには、ポリシー管理者とエージェント所有者の両方が設定プロセスを実行する必要があります。 このシナリオでは、HTML フォーム認証のサポートが 1 つ以上のエージェントで必要であることをポリシー管理者がユーザに伝えるときにエージェント所有者が実行する必要があるプロセスを説明します。

注: ポリシー管理者が HTML フォーム認証を設定する方法の詳細については、関連シナリオ「*HTML フォーム認証の設定方法*」を参照してください。



1. 基本的な FCC 認証の設定 (P. 199)
2. Domino Web サーバでの FCC リダイレクト用 URL のマップ (P. 204)
3. POST 維持の設定 (P. 204)
4. 高度な FCC 設定
5. FCC のパフォーマンスの調整 (P. 226)

基本的な FCC の操作の設定

HTML フォーム認証方式によって保護されるリソースを保護するエージェントの *フォーム クレデンシャル コレクタ* (*FCC*) コンポーネントを設定する基本的な設定手順を実行します。

1. IIS Web サーバまたは Domino Web サーバを使用している場合は、FCC 用の MIME タイプ マッピングを設定 (P. 199) します。

   注: エージェント設定ウィザードは、SiteMinder 認証情報コレクタが以下のタイプの Web サーバに使用する適切な MIME 形式を自動的にセットアップします。

   - Apache Web サーバおよび Apache ベースの Web サーバ。
   - Oracle iPlanet Web サーバ

2. FCC で使用するために、エージェント ID と Web サーバをマッピングします。
3. 以下の追加設定項目を必要に応じて設定します。
   - FCC/SCC による完全修飾ホスト名としてのエージェント名の使用を有効化 (P. 200)。
   - シングル リソース ターゲットを使用するための FCC の設定 (P. 200)。
   - 認証情報コレクタのリダイレクトでの相対ターゲットの使用 (P. 201)。
   - 有効なターゲット ドメインの定義 (P. 89)。
   - 有効なフェデレーション ターゲット ドメインの定義 (P. 202)。

IIS および Domino Web サーバで FCC 用の MIME タイプ マッピングを設定する

IIS および Domino Web サーバで、FCCExt エージェント設定パラメータを指定して、Web エージェント設定に FCC 用の MIME タイプ マッピングを設定します。 MIME タイプ マッピングはファイル拡張子として表されます。デフォルト値の使用をお勧めします。

FCCExt

FCC 用の MIME タイプ マッピングを指定します。

デフォルト： .fcc

制限： 有効なファイル拡張子。

例： .myfcc

注: デフォルトの拡張子を使用しない場合、またはデフォルト値がすでに使用されている場合は、代わりに任意の拡張子を入力してください。 たとえば、FCC に対応する FCCExt を .myfcc に設定し、その拡張子を使用するように FCC テンプレートの名前を変更（たとえば login.myfcc）した場合、エージェントは .myfcc で終わる URL を、フォーム認証リクエストとして認識します。

FCC/SCC による完全修飾ホスト名としてのエージェント名の使用の有効化

フォームおよび SSL のクレデンシャル コレクタでターゲット URL の完全修飾ホスト名をエージェント名として使用できるようにするには、AgentNamesAreFQHostNames 設定パラメータを定義します。

たとえば、AgentNamesAreFQHostNames パラメータを Yes に設定すると、次の URL 文字列の www.nete.com 部分が Web エージェント名になります。

```
url?A=1&Target=http://www.nete.com/index.html
```

クレデンシャル コレクタは次の状況でこのパラメータを使用します。

- ターゲット エージェントが URL にエージェント名を追加しない場合（サード パーティのエージェントで発生することがあります）。
- AgentName パラメータ内で、エージェントからホスト名へのマッピングを設定しなかった場合。

AgentNamesAreFQHostNames パラメータが no に設定されている場合、認証情報コレクタは DefaultAgentName パラメータの値を、ターゲット Web エージェントの名前として使用します。

シングル リソース ターゲットを使用するための FCC の設定

ユーザをシングル リソースにダイレクトするよう FCC を設定するには、テンプレート ファイル login.fcc の中でターゲットをハード コード化します。

次の手順に従ってください:

1. *agent_home*/Samples にある login.fcc ファイルを開きます。
2. @target=*target_resource* を FCC に追加します。
3. 次のエントリを追加します。

   @smagentname=*agent_name_protecting_resource*

   例： @smagentname=mywebagent

4. EncryptAgentName パラメータを no に設定します。エージェント名をファイル内でハードコード化した後、そのエージェント名を暗号化する手法は存在しないので、このパラメータが必要です。
5. この FCC を使用する他のすべてのエージェントに対して、EncryptAgentName を no に設定します。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

認証情報コレクタのリダイレクトでの相対ターゲットの使用

必要に応じて、要求を認証情報コレクタとターゲット リソースへダイレクトする際に、完全修飾 URL の代わりに相対 URI を使用するようエージェントに指示します。 相対 URI を使用すると、Web エージェントと共に他のシステム上に存在しているクレデンシャル コレクタがリクエストを処理することを防止できます。

注: この設定項目は、cookie 認証情報コレクタ（CCC）を*除く*、他のすべての認証情報コレクタに適用されます。 CCC は、このパラメータの完全修飾ドメイン名を使用する必要があります。 相対 URI を使用した場合、OnAuthAccept レスポンスは CCC で適切に動作しません。

通常、完全修飾 URL が認証情報コレクタの URL に付加されます。 例：

url?A=1&Target=http://www.nete.com/index.html

相対 URI のみを使用するには、TargetAsRelativeURI パラメータを yes に設定します。 yes に設定した場合、認証情報コレクタの URL に付加されるターゲット パラメータは相対ターゲット（url?A=1&Target=/index.html など）になります。その場合、認証情報コレクタがリダイレクトを行って、ターゲットリソースを保護している Web エージェントへ戻すときは、相対リダイレクトになります。 また、Web エージェントは、スラッシュ（/）以外の文字で始まるすべてのターゲットを拒否します。

このパラメータのデフォルト値は no なので、常に完全修飾 URL が使用されます。

有効なターゲット ドメインの定義

悪意のある Web サイトにユーザをリダイレクトするフィッシング攻撃からリソースを保護するように SiteMinder エージェントを設定するには、以下の設定パラメータを設定します。

ValidTargetDomain

認証情報コレクタがユーザをリダイレクトすることを許可されるドメインを指定します。 URL 内のドメインがこのパラメータ内で設定されたドメインに一致しない場合は、リダイレクトが拒否されます。

デフォルト： No

フォーム クレデンシャル コレクタ（FCC）を含むすべての高度な認証方式が、このパラメータをサポートします。

ValidTargetDomain パラメータは、処理の間にターゲットの有効なドメインを識別します。 ユーザをリダイレクトする前に、エージェントはリダイレクト URL の値と、このパラメータ内のドメインを比較します。 このパラメータがない場合、エージェントはユーザをどのようなドメイン内のターゲットにもリダイレクトさせます。

ValidTargetDomain パラメータには、有効なドメインごとに 1 つの値を設定し、複数の値を含めることもできます。

ローカル Web エージェント設定では、以下の例のように、各ドメインについて、1 行に 1 つのエントリを指定します。

```
validtargetdomain=".xyzcompany.com"

validtargetdomain=".abccompany.com"
```

有効なフェデレーション ターゲット ドメインの定義

SiteMinder が レガシー フェデレーション SP として動作している場合、SAML 2.0 トランザクション用に Identity Provider Discovery （IPD）プロファイルを設定できます。 IPD によって、認証リクエストに対してどの IdP がアサーションを生成するかをユーザが選択できるようになります。

検出プロセスで、悪意のある Web サイトにユーザがリダイレクトされるのを防ぐことができます。認証リクエストを満たす IdP のドメインが検証されるよう Web エージェントを設定します。

検証プロセスを有効にするには、以下のパラメータの値を設定します。

ValidFedTargetDomain

（Federation のみ-SAML 2.0） Identity Provider Discovery を実装した場合に、フェデレーション環境の有効なドメインをすべてリスト表示します。

SiteMinder Identity Provider Discovery （IPD）サービスでリクエストを受信すると、リクエストの IPDTarget クエリ パラメータを調べます。 このクエリ パラメータは、Discovery サービスでリクエストを処理した後にリダイレクトする URL をリスト表示します。IdP の場合、IPDTarget は SAML 2.0 シングル サインオン サービスです。SP の場合、ターゲットは共通ドメイン cookie を使用するリクエスト アプリケーションです。

フェデレーション Web サービスでは、IPDTarget URL のドメインを、ValidFedTargetDomain パラメータに指定されたドメインのリストと比較します。URL ドメインが ValidFedTargetDomain に設定されたドメインの 1 つと一致する場合、IPD サービスは IPDTarget パラメータに示された URL にユーザをリダイレクトします。 このリダイレクトは SP の URL に対して行われます。

ドメインが一致しない場合、IPD サービスはユーザ リクエストを拒否し、ブラウザに 403 Forbidden が返されます。 また、FWS トレース ログおよび affwebservices ログにエラーが報告されます。 これらのメッセージは、IPDTarget のドメインが有効なフェデレーション ターゲット ドメインとして定義されないことを示します。

ValidFedTargetDomain を設定しない場合、検証は行われず、ユーザはターゲット URL にリダイレクトされます。

制限： フェデレーション ネットワーク内の有効なドメイン

デフォルト： デフォルトなし

ValidFedTargetDomain パラメータに有効なドメインを指定します。 この設定は複数パラメータなので、複数のドメインを入力できます。

ローカル設定ファイルを変更している場合は、たとえば以下のように、ドメインを別々にリスト表示します。

validfedtargetdomain=".examplesite.com"

validfedtargetdomain=".abccompany.com"

Identity Provider Discovery プロファイルの詳細については、「Federation Security Services Guide」を参照してください。

Domino Web エージェントによる FCC リダイレクト用 URL のマップ

フォーム認証方式を使用して Domino ビュー（.nsf）のリソースを保護するには、フォーム認証情報コレクタにリダイレクトする前に、URL をマップする必要があります。

次の手順に従ってください:

1. DominoNormalizeUrls パラメータの値を yes に設定します。
2. DominoMapUrlForRedirect パラメータの値を yes に設定します。

   Domino の URL は FCC へリダイレクトされる前にマップされます。

POST 維持の設定

SiteMinder は、ユーザが FCC フォームにポストするデータを自動的に維持します。 この維持メカニズムは、タイムアウトまたは他の障害が POST 操作中に発生した場合に、フォーム上のデータの損失を防ぎます。

環境で従来型エージェントとフレームワーク エージェントを組み合わせて使用している場合は、次の追加の設定手順が必要です。

- フレームワーク エージェントと従来のエージェントの間の POST 維持の有効化（P. 205）
- POST 維持ページのカスタマイズ（P. 206）

POST 維持を使用しない場合は、これを無効化 (P. 100)できます。

POST 維持は以下の状況ではサポートされません。

- ACE 認証
- FCC へポストする、任意のカスタム認証方式

フレームワーク エージェントと従来のエージェントの間の POST 維持の有効化

フレームワーク エージェントは POST 維持データを従来のエージェントとは異なる方法で処理します。SiteMinder 環境でフレームワーク エージェントと従来のエージェントが組み合わせて使用され、一方のタイプのエージェントによってホストされているリソースがもう一方のタイプのエージェントでホストされているフォーム認証情報コレクタ（FCC）によって保護される場合、以下のパラメータを備えた適切なテンプレート ファイルを指定する必要があります。

PostPreservationFile

以下の POST 維持テンプレート ファイルの*いずれか*に対するパスを指定することで、トラディショナル エージェントとフレームワーク エージェントとの間の POST 維持データの転送を有効にします。

- tr2fw.pptemplate - トラディショナル エージェントが稼働しているサーバでホストされているリソースが、フレームワーク エージェント上で実行されている FCC によって保護されていることを示します。
- fw2tr.pptemplate - フレームワーク エージェントが稼働しているサーバでホストされているリソースが、トラディショナル エージェント上で実行されている FCC によって保護されていることを示します。

デフォルト： デフォルトなし

例： *web_agent_home*/samples/forms/fw2tr.pptemplate

フレームワーク エージェントと従来のエージェントの間の POST 維持を有効にする方法

1. 別のタイプのエージェント上で実行されている FCC によって保護されるリソースを決定します。
   a. フレームワーク エージェントで実行されている FCC によって保護されるリソースをホストしている従来のエージェントのリストを作成します。
   b. 従来のエージェントで実行されている FCC によって保護されるリソースをホストしているフレームワーク エージェントのリストを作成します。
2. リソースをホストしている従来のエージェント（事前に手順 1a で列挙したもの）で、PostPreservationFile パラメータの値を tr2fw.pptemplate ファイルのパスに設定します。

3. リソースをホストしているフレームワーク エージェント（事前に手順 1b で列挙したもの）で、PostPreservationFile パラメータの値を fw2tr.pptemplate ファイルのパスに設定します。
4. 従来のエージェントと通信するフレームワーク Web エージェントのすべてで、以下のパラメータの値を yes に設定します。

   LegacyPostPreservationEncoding

   Web エージェントが POST 維持データをエンコードするときに、以前のトラディショナル Web エージェントと互換性のある方法を使用するか、新規のフレームワーク Web エージェントと互換性のある方法を使用するかを指定します。 このパラメータの値が yes の場合は、トラディショナル Web エージェントと互換性のあるエンコーディングが使用されます。このパラメータの値が no の場合は、フレームワーク Web エージェントのみと互換性のあるエンコーディングが使用されます。

   デフォルト： No
5. ユーザのリソースをホストしている Web サーバを再起動します。

   フレームワーク エージェントと従来のエージェントの間の POST 維持が有効になります。

POST 維持ページのカスタマイズ

POST 操作中にタイムアウトまたは他の障害が発生した場合は、POST 維持ページが表示されます。 ほとんどの場合、2 ページ未満の POST 維持ページが表示されます。 ただし、ポストされているフォーム データの量が多い場合は、5 秒間 POST 維持ページを表示できます。

デフォルトでは、POST 維持ページは以下のテキストを表示します。

このページは、ユーザが要求に対して許可される間、ユーザのデータを保持するために使用されます。ユーザは許可プロセスを続行するために転送されます。 これが自動的に発生しない場合は、下の［続行］ボタンをクリックします。

POST 維持ページは、ユーザに対してアプリケーションにデータを再ポストすることを許可する［続行］ボタンも表示します。

POST 維持ページをカスタマイズするには、POST 維持テンプレート ファイルを作成します。

デフォルト ページの一般的な構造を以下に示します。

```
<HTML><HEAD><TITLE></TITLE></HEAD><BODY onLoad="document.AUTOSUBMIT.submit();">
このページは、ユーザが要求に対して許可される間、ユーザのデータを保持するために使用されます。
<BR><BR>
ユーザは許可プロセスを続行するために転送されます。 これが自動的に発生しない場合は、下の［続行］ボタンをクリックします。
<FORM NAME="AUTOSUBMIT" METHOD="POST" ACTION="$$smpostlocation$$">
<$$smpostdata$$>
<INPUT TYPE="SUBMIT" VALUE="Continue">
</FORM></BODY></HTML>
```

POST 維持テンプレートには、POST 維持ページをレンダリングするときに Web エージェントが展開する、次の 2 つのエレメントが含まれる必要があります。

$$smpostlocation$$

POST 維持の第 1 段階で、認証情報コレクタ URL まで展開されます。
POST 維持の第 2 段階で、保護リソース の URL まで展開されます。

$$smpostdata$$

それぞれの POST 維持の段階に、いずれか一方の場所にポストされる正しいフォーム データをもたらす結果になる HTML が含まれるまで展開されます。

これらのエレメントは削除、あるいは変更しないでください。

ただし、他のエレメントは変更できます。 たとえば、［続行］ボタンを削除するには、そのボタンを定義する <INPUT> エレメントを削除します。

```
<INPUT TYPE="SUBMIT" VALUE="Continue">
```

2 つのサンプル POST 維持テンプレート ファイル、fw2tr.pptemplate および tr2fw.pptemplate が以下の場所に含まれています。

- **UNIX**： *web_agent_home*/samples_default/forms/
- **Windows**： *web_agent_home*¥samples_default¥forms¥

  web_agent_home

  Web サーバに Web エージェントがインストールされるディレクトリを示します。

POST 維持テンプレート ファイルを使用するように Web エージェントを設定するには、PostPreservationFile エージェント設定パラメータを定義して、テンプレート ファイルのパスを指定します。
例：

```
PostPreservationFile="/app/netegrity/webagent/samples_default/forms/nosubmitbutton.pptemplate"
```

POST 維持の無効化

POST 維持を使用する必要がない場合は、以下のパラメータを使用してそれを無効にすることができます。

PreservePostData

要求をリダイレクトする場合に Web エージェントが POST データを維持するかどうかを指定します。 ユーザが、フォーム認証や証明書認証など、高度な認証を受ける場合、POST データは認証フェーズの間、維持されます。

デフォルト： yes

POST 維持を無効にするには、PreservePostData パラメータの値を no に設定します。

高度な FCC 設定

ニーズに合わせて以下の高度な認証情報コレクタ設定が可能です。

- 小文字を使用した URL プロトコル部分の指定 (P. 209)。
- フォームによる SafeWord ユーザの認証。
- ACE ユーザの認証。
- リダイレクト URL のクエリ文字列の暗号化 (P. 210)。
- リダイレクト URL のクエリ文字列をエンコードするための FCC ディレクティブ (P. 211)。
- Windows 認証を許可するように FCC を設定 (P. 215)。
- FCC でのアプリケーション リクエスト ルーティングの使用 (P. 212)。

小文字でのリダイレクト URL プロトコルの指定

RFC 2396 に準拠しないレガシー アプリケーションをフォーム ベース認証方式で保護する場合、URL のプロトコル部分を小文字にする必要があれば、以下のパラメータを設定します。

LowerCaseProtocolSpecifier

リダイレクト URL のスキーム（プロトコル）部分で小文字のみを使用するかどうかを指定します。 この設定パラメータは、RFC 2396 に準拠していないレガシー アプリケーションに対応します。 この RFC には、アプリケーションは URL のプロトコル部分で大文字と小文字の両方を処理する必要があると記載されています。 以下のいずれかの状況でこのパラメータを変更します。

- RFC 2396 に準拠していないレガシー アプリケーションを使用する場合。
- リダイレクト URL にクエリ データが含まれている場合。
- HTML フォーム（FCC）認証方式を使用する場合。

デフォルト： No （HTTP、HTTPS のように大文字を使用）

例： Yes （http、https のように小文字を使用）

お使いの環境で URL に小文字のプロトコルを指定するには、LowerCaseProtocolSpecifier パラメータの値を yes に設定します。

リダイレクト URL 内のクエリ文字列パラメータの暗号化

以下のパラメータにより、Web エージェントはリダイレクト URL 内のすべての SiteMinder クエリ パラメータを暗号化できます。

SecureURLs

Web エージェントがリダイレクト URL 内の SiteMinder クエリ パラメータを暗号化するかどうかを指定します。この設定を使用して、高度な認証方式であるパスワード サービスによって保護されている要求されたリソースのセキュリティを強化したり、要求が cookie プロバイダを呼び出すときのセキュリティを強化したりすることができます。

重要：Web エージェントは、SiteMinder コンポーネント間で送信されたデータを暗号化するだけです。 リダイレクトのために SiteMinder 以外のアプリケーションに送信されるデータは暗号化されません。

以下の SiteMinder 認証情報コレクタおよびアプリケーションは SecureUrls 機能をサポートします。

- HTML フォーム認証
- 証明書およびフォーム認証
- SSL 認証
- 証明書またはフォーム認証
- NTLM 認証
- ACE 認証
- SafeWord 認証
- ユーザによる自己登録
- cookie プロバイダによるマルチドメイン シングル サインオン
- FCC ベースのパスワード サービス（CGI または JSP ベースではない）

デフォルト： No

次の手順に従ってください:

1. SecureURLs パラメータの値を yes に設定します。
2. シングル サインオン環境内のリダイレクト URL 内のクエリ文字列パラメータを暗号化するには、シングル サインオン環境内のすべての Web エージェントの SecureURL パラメータが同じ値に設定されていることを確認します。
3. カスタム FCC を使用している場合、他の FCC ディレクティブ（TARGET など）と共に、smquerydata ディレクティブをカスタム FCC に追加します。

   クエリ文字列パラメータは SiteMinder リダイレクト URL 内で暗号化されます。

リダイレクト URL のクエリ文字列をエンコードするための FCC ディレクティブ

認証情報コレクタのリダイレクト URL のクエリ文字列は暗号化できます。認証情報コレクタは、クエリ データを暗号化するために使用されるキーを提供します。

フォーム認証方式の場合、クエリ文字列ディレクティブ smquerydata は FCC テンプレートの一部です。 FCC を提供するエージェントは、FCC がポスティングされると、このディレクティブを使用して、暗号化されたクエリ データを目的のエージェントに送信します。

使用されるディレクティブは、以下のとおりです。

```
<INPUT type='hidden' name='smquerydata' value='$$smquerydata$$>
```

注: カスタム FCC を使用している場合は、他の FCC ディレクティブ（TARGET など）と共に、smquerydata ディレクティブをカスタム FCC に追加します。

SiteMinder 12.52 エージェントは、SecureUrls パラメータが有効である場合、この機能をサポートする他のエージェントから提供される認証情報コレクタとのみ連携できます。

HTML フォーム認証でアプリケーション リクエスト ルーティング（ARR）を設定する方法

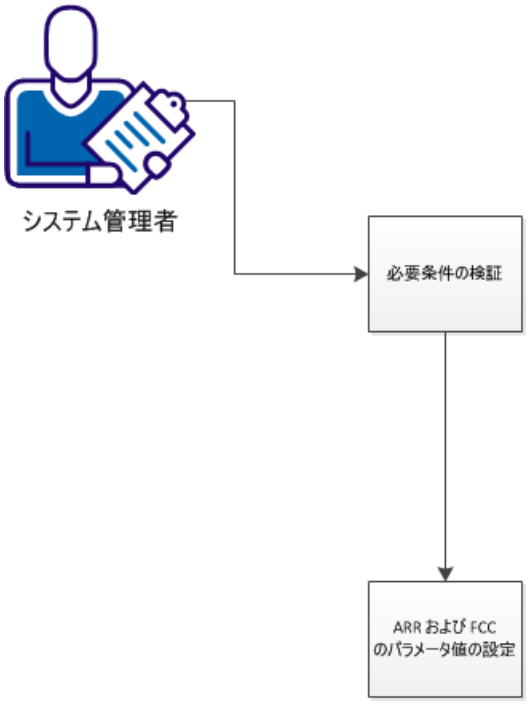
アプリケーション リクエスト ルーティング（ARR）は Microsoft Internet Information Services （IIS）で利用可能なオプション機能です。 ARR はちょうどプロキシ サーバのように、他のサーバに要求を転送します。

IIS Web サーバは、ARR を使用して Cookie を違う方法で処理します。 この設定は、FCC 認証方式において SiteMinder Cookie が処理される方法に影響します。

このシナリオでは、以下のいずれかの状況において <stmdnr> エージェントが必要とする追加の構成設定について説明します。

- ARR は FCC と併用される。
- ARR は SiteMinder および Arcot で使用される。

次の図は、システム管理者が FCC を使用して ARR 用に SiteMinder を設定する方法を説明しています。

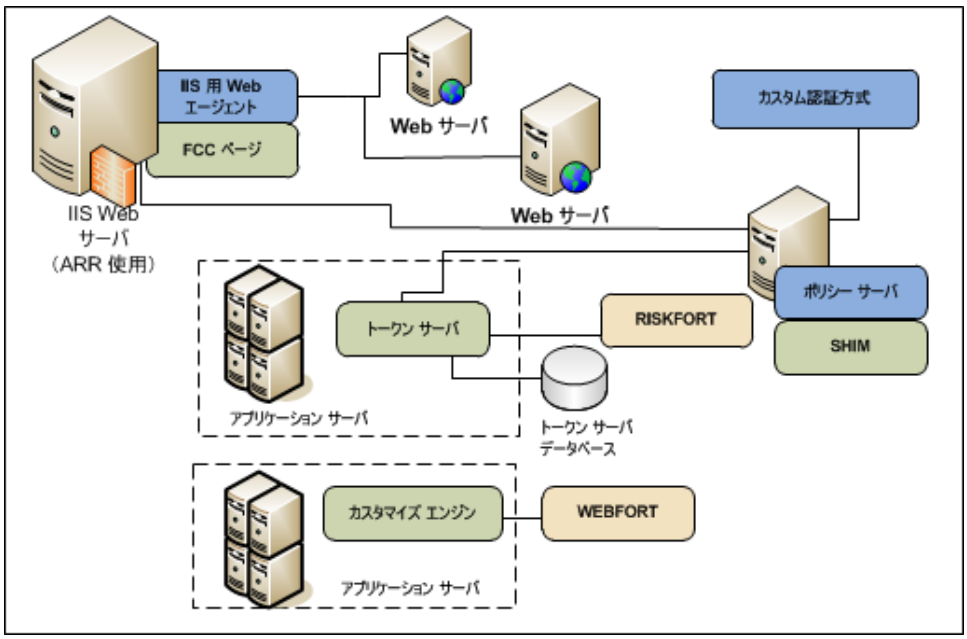


ARR と FCC を併用する SiteMinder エージェントを設定するには、次の手順に従ってください：

1. ユーザの環境がその前提条件を満たすことを確認します (P. 213)。
2. ARR と FCC のパラメータ値を設定します (P. 214)。

前提条件の確認

以下の図では、ユーザ環境のコンポーネントおよび前提条件について説明しています。



ユーザの SiteMinder および [assign the value for dlp in your book] 環境が以下の要件を満たすことを確認します。

- 以下のコンポーネントと共に、完全な SiteMinder 環境がインストールされ設定されている。
  - ARR を実行する IIS Web サーバの背後で展開する Web サーバ上のリソースを保護するポリシー。
  - （オプション）インストールされており設定されている CA Arcot コンポーネント。
- 以下のアイテムを持つ IIS Web サーバがインストールされ設定されている。
  - 要求を Web サーバに転送するように設定されたアプリケーション リクエスト ルーティング（ARR）。
  - ARR が実行されているサーバ上にインストールされ設定されている IIS 用 SiteMinder エージェント。
  - FCC フォーム認証方式。
- エージェントが中央設定を使用するかローカル設定を使用するかを決定します。

ARR と FCC のパラメータ値の設定

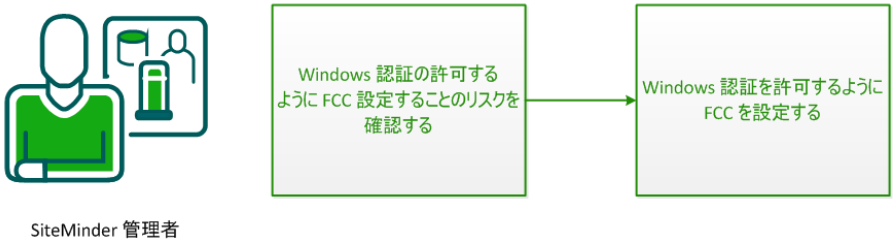
ARR および FCC のパラメータ値を設定するには、次の手順に従ってください：

1. エージェント設定メソッドに相当する以下のリストのタスクを実行します。
   - 中央設定の場合は、エージェント設定オブジェクトを開きます (P. 34)。
   - ローカル設定の場合は、Web サーバ上のローカル設定ファイルを開きます (P. 40)。
2. FCCCompatMode パラメータを見つけて、その値を yes に変更します。
3. CookieDomain パラメータを見つけて、その値を none に変更します（値は空白のままにしておかないでください）。

Windows 認証を許可するように FCC を設定する方法

SiteMinder フォーム認証情報コレクタ（FCC）は、CA サービスによりカスタム認証方式を安全にトリガできるように設計されています。 そのため、FCC は、任意の認証方式についてユーザを認証できます。 ただし、FCC はデフォルトでは Windows 認証方式に対して認証 *しません*。 この動作により、ある設定内で任意の有効な Windows ユーザの SiteMinder セッションを生成するために、攻撃者が FCC を利用するのを防ぐことができます。

環境で FCC による Windows 認証方式での認証が必要な場合は、EnableFCCWindowsAuth エージェント設定パラメータを指定することにより、これを有効にできます。 ただし、Windows 認証に対する FCC のサポートを有効にする前に、そのリスクを確認し、脆弱性が露出される設定に注意してください。



1. FCC による Windows 認証の許可を有効にすることによるリスクを確認します (P. 215)。
2. Windows 認証を許可するように FCC を設定します (P. 217)。

FCC による Windows 認証の許可を有効にすることによるリスク

デフォルトでは、FCC は Windows 認証方式で認証を行いません。 ユーザは FCC による Windows 認証の許可を有効にできます。 ただし、そうすることにより、攻撃者がある設定内で任意の有効な Windows ユーザの SiteMinder セッションを生成するために FCC を使用する恐れがあるという脆弱性が露呈します。

この脆弱性は、HTML フォームで保護されるレルムと Windows で保護されるレルムの両方で、同じ SiteMinder エージェント名またはエージェント グループ名が使用される設定で出現します。たとえば HTML フォーム認証および Windows 認証で設定されたそれぞれ別のレルムを保護するように、単一の Web エージェントが設定されている場合です。

以下のシナリオ例について考えてみます。

- リソース A は HTML フォーム認証を使用して保護されたレルムに設定されています。 FCC は HTML フォームにより、リソース A にアクセスするユーザを認証します。
- リソース B は Windows 認証を使用して保護されたレルムに設定されています。リソース B にアクセスするユーザは Windows 認証を完了しています。
- 両方のリソースは同じ IIS サーバ上でホストされ、同じ Web エージェントによって保護されます。 そのため、両方のレルムは同じエージェント名で設定されます。

攻撃は次のように発生します。

1. 攻撃者は HTML フォーム内の TARGET パラメータを、「リソース A」から「リソース B」に変更します。
2. 攻撃者は、任意の有効な Windows ユーザ名を使ってこのフォームをサブミットします。
3. FCC はユーザ名を認証のためにポリシー サーバへ渡します。 SiteMinder は、HTML フォーム認証方式の代わりに Windows 認証方式を実行し、ユーザ名が検証されます。

この結果、SiteMinder セッションはユーザに戻され、新しいセッションが有効であると見なされる場合には、後続のすべての要求に対するシングルサインオンが可能になります。攻撃者は、その Windows ユーザ名が FCC にサブミットされたユーザを偽装するのです。

Windows 認証を許可するように FCC を設定

以下のエージェント設定パラメータを指定することにより Windows 認証を許可するように FCC を設定します。

EnableFCCWindowsAuth

FCC として動作するエージェントが SiteMinderWindows 認証方式が保護するリソースに対してユーザを認証できるかどうかを指定します。

このパラメータには、以下の値を設定できます。

- Yes - FCC は Windows 認証方式で認証できます。

  重要: このパラメータが Yes に設定されている場合、攻撃者は必要な認証情報を提供せずに、FCC を利用して Windows ユーザを偽装する可能性があります。

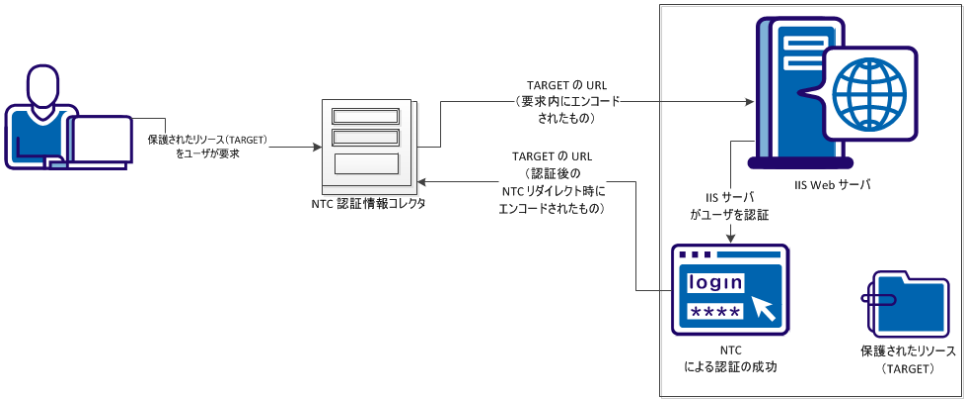
- No - FCC は Windows 認証方式で認証できません。

デフォルト： No

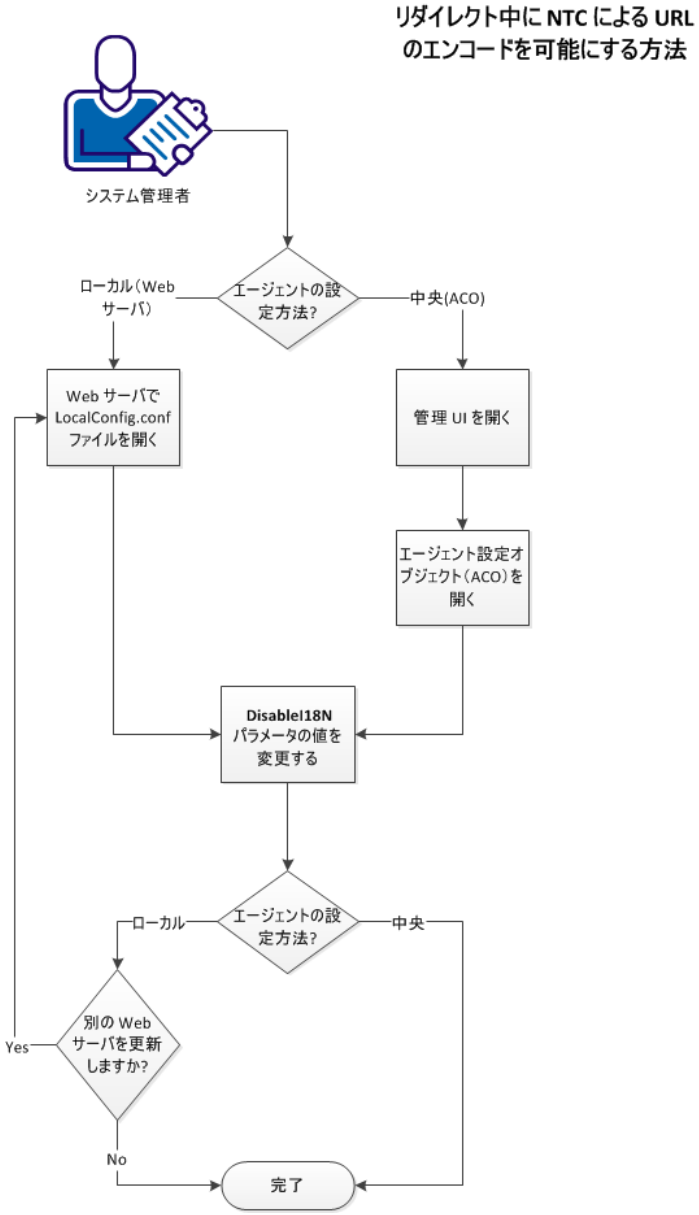
保護されたリソースへのリダイレクト中に NTC による URL のエンコードを可能にする方法

SiteMinder は Windows 認証情報コレクタ（NTC）を使用して、リソースを保護できます。ユーザは NTC へ認証情報をサブミットし、次に、NTC は IIS Web サーバにユーザをログインさせます。 IIS Web サーバがユーザを認証します。 NTC は認証後にユーザを保護されている（TARGET）リソースにリダイレクトします。

NTC は通常、要求中に URL の TARGET 部分内の文字をエンコードしますが、それは認証後のリダイレクト中ではありません。 URL の TARGET 部分がリダイレクト中にエンコードされるように、エージェント設定を変更できます。 以下の図では、この動作について説明しています。



以下の図は、保護されているリソースの要求中に NTC による URL のエンコードを可能にするプロセスを示しています。



保護リソースへのリダイレクト中に NTC による URL のエンコードを可能にするには、次の手順に従ってください：

1. 以下のリストからユーザのエージェント設定メソッドに一致する手順を選択します。
   - ポリシー サーバ上でエージェント設定オブジェクト（ACO）を使用するエージェントについては、以下の手順に従います。
     a. 管理 UI を開きます (P. 220)。
     b. エージェント設定オブジェクト（ACO）を開き、次に、DisableI18N パラメータの値を変更します (P. 221)。
   - Web サーバ上でローカル設定ファイルを使用するエージェントについては、以下の手順に従います。
     a. テキスト エディタを使用して Web サーバ上の LocalConfig.conf ファイルを開き、次に、DisableI18N パラメータの値を変更します (P. 224)。
2. ローカル設定を使用するエージェントについては、各 Web サーバについて手順 1c を繰り返します。

   NTC は保護されているリソースへのリダイレクト中にエンコードされた URL を使用します。

管理 UI でのポリシー サーバ オブジェクトの変更

管理 UI を開いて、ポリシー サーバ上のオブジェクトを変更します。

次の手順に従ってください:

1. ブラウザで以下の URL を入力します。

   `https://host_name:8443/iam/siteminder/adminui`

   host_name

   管理 UI ホスト システムの完全修飾名を指定します。

2. ［ユーザ名］フィールドに SiteMinder スーパーユーザ名を入力します。
3. ［パスワード］フィールドに SiteMinder スーパーユーザ アカウントのパスワードを入力します。

   注: スーパーユーザ アカウントのパスワードにドル記号（$）文字が含まれている場合は、ドル記号文字の各インスタンスを $DOLLAR$ に置換します。たとえば、SiteMinder スーパーユーザ アカウント パスワードが $password の場合は、パスワード フィールドに「$DOLLAR$password」と入力します。

4. 適切なサーバ名または IP アドレスが［サーバ］ドロップダウン リストに表示されていることを確認します。
5. ［ログイン］を選択します。

エージェント設定オブジェクトの DisableI18N パラメータ値の変更

中央で設定された Web エージェントのターゲット URL 内の HTTP エンコード文字を処理するように Windows 認証情報コレクタを設定できます。中央で設定された Web エージェントは、ポリシー サーバ上のエージェント設定オブジェクトに格納されたパラメータ設定を使用します。

次の手順に従ってください:

1. ［インフラストラクチャ］-［エージェント設定オブジェクト］をクリックします。

   エージェント設定オブジェクトのリストが表示されます。

   目的の［エージェント設定オブジェクト］の行で編集アイコンをクリックします。

   ［エージェント設定の変更］ダイアログ ボックスが表示されます。

2. 次のパラメータの左側の編集アイコンをクリックします。

   DisableI18N

   TARGET URL の文字が HTTP エンコーディングを使用している場合に、Windows 認証情報コレクタ（NTC）が認証中に TARGET URL をどのように処理するかを指定します。このパラメータの値が *No* の場合、URL 内のすべての文字が認証中にデコードされます。デコードされた文字は、TARGET リソースへのリダイレクトで使用されます。 このパラメータの値が *Yes* の場合、TARGET URL 内の文字は認証中にデコードされません。 HTTP エンコーディングを使用するすべての文字は、認証前および認証後もエンコードされたままとなります。

   デフォルト： No

   ［パラメータの編集］ダイアログ ボックスが表示されます。

3. ［値］フィールドのテキストを yes に変更します。

4. ［OK］をクリックします。

   ［パラメータの編集］ダイアログ ボックスが閉じ、［エージェント設定の変更］ダイアログ ボックスが表示されます。

5. 次のパラメータの左側の編集アイコンをクリックします。

   BadUrlChars

   URL リクエストに使用できない文字シーケンスを指定します。 Web エージェントはこのパラメータ内のリストに対して、「?」文字の前にある URL 内の文字を確認します。 指定された文字のいずれかが見つかった場合、Web エージェントは要求を拒否します。

   以下の文字を指定できます。

   - 円記号（¥）
   - ダブルスラッシュ（//）
   - ピリオドとスラッシュ（./）
   - スラッシュとピリオド（/.）
   - スラッシュとアスタリスク（/*）
   - アスタリスクとピリオド（*.）
   - ティルダ（~）
   - %2d
   - %20
   - %00-%1f
   - %7f-%ff
   - %25

   複数の値はカンマで区切ります。 スペースは*使用しないでください*。

   無効な URL 文字は、その前に疑問符 (?) が付いている場合にのみ、CGI パラメータの中で使用できます。

デフォルト： 無効（すべての文字が許可されます）。

制限：

- デフォルトの 16 進数が英語の文字に適用されます。 その他の言語の場合、許可する言語の文字に対応する 16 進数値を削除します。 対象となる言語（ただしこれらに限定されません）には、ポルトガル語（ブラジル）、フランス語、日本語、および中国語などがあります。
- 文字は実際にその文字を入力して指定できます。 さらに、その文字の URL エンコード形式を入力することもできます。たとえば、文字 a を入力するか、または、そのエンコード値である %61 を入力できます。
- 最大 4096 文字まで指定できます（区切り文字として使用するカンマを含みます）。
- 文字の範囲をハイフンで区切って指定することもできます。構文は *starting_character-ending_character* です。 たとえば、一連の文字として「a-z」と入力できます。
- 引用符（"）を URL エンコード値 %22 で指定します。 ASCII は使用できません。

［パラメータの編集］ダイアログ ボックスが表示されます。

6. ［値］フィールドから以下のテキストを削除します。

   ,%25

7. ［OK］をクリックします。

   ［パラメータの編集］ダイアログ ボックスが閉じ、［エージェント設定の変更］ダイアログ ボックスが表示されます。

8. ［サブミット］をクリックします。

   ［エージェント設定の変更］ダイアログ ボックスが閉じ、確認メッセージが表示されます。

9. （オプション）将来の参照用に、変更に関する任意の注釈を［コメント］フィールドに入力します。

10. ［はい］をクリックします。

    変更は、次回 Web エージェントがポリシー サーバをポーリングしたときに適用されます。

LocalConfig.conf ファイル内の DisableI18N パラメータ値の変更

ターゲット URL 内の HTTP エンコード文字を処理するように Windows 認証情報コレクタを設定できます。 ローカルで設定された Web エージェントは、各 Web サーバ上の設定ファイルに格納されたパラメータ設定を使用します。

次の手順に従ってください:

Web サーバ上で LocalConfig.conf ファイルを見つけます。次のリストの例を使用して、ご使用のタイプの Web サーバ上でファイルを見つけます。

IIS Web サーバ

web_agent_home¥bin¥IIS

Oracle iPlanet Web サーバ

Oracle_iPlanet_home/https-hostname/config

Apache Web サーバ

Apache_home/conf

1. テキスト エディタを使用して LocalConfig.conf ファイルを開き、次に、以下のパラメータを見つけます。

   DisableI18N

   TARGET URL の文字が HTTP エンコーディングを使用している場合に、Windows 認証情報コレクタ（NTC）が認証中に TARGET URL をどのように処理するかを指定します。このパラメータの値が *No* の場合、URL 内のすべての文字が認証中にデコードされます。デコードされた文字は、TARGET リソースへのリダイレクトで使用されます。 このパラメータの値が *Yes* の場合、TARGET URL 内の文字は認証中にデコードされません。 HTTP エンコーディングを使用するすべての文字は、認証前および認証後もエンコードされたままとなります。

   デフォルト： No

2. DisableI18n パラメータの値を yes に変更します。

3. 以下のパラメータを探します。

   BadUrlChars

   URL リクエストに使用できない文字シーケンスを指定します。 Web エージェントはこのパラメータ内のリストに対して、「?」文字の前にある URL 内の文字を確認します。 指定された文字のいずれかが見つかった場合、Web エージェントは要求を拒否します。

   以下の文字を指定できます。

   - 円記号（¥）
   - ダブルスラッシュ（//）
   - ピリオドとスラッシュ（./）
   - スラッシュとピリオド（/.）
   - スラッシュとアスタリスク（/*）
   - アスタリスクとピリオド（*.）
   - ティルダ（~）
   - %2d
   - %20
   - %00-%1f
   - %7f-%ff
   - %25

   複数の値はカンマで区切ります。 スペースは*使用しないでください*。

   無効な URL 文字は、その前に疑問符 (?) が付いている場合にのみ、CGI パラメータの中で使用できます。

デフォルト： 無効（すべての文字が許可されます）。

制限：

- デフォルトの 16 進数が英語の文字に適用されます。 その他の言語の場合、許可する言語の文字に対応する 16 進数値を削除します。 対象となる言語（ただしこれらに限定されません）には、ポルトガル語（ブラジル）、フランス語、日本語、および中国語などがあります。
- 文字は実際にその文字を入力して指定できます。 さらに、その文字の URL エンコード形式を入力することもできます。たとえば、文字 a を入力するか、または、そのエンコード値である %61 を入力できます。
- 最大 4096 文字まで指定できます（区切り文字として使用するカンマを含みます）。
- 文字の範囲をハイフンで区切って指定することもできます。構文は *starting_character-ending_character* です。 たとえば、一連の文字として「a-z」と入力できます。

引用符（"）を URL エンコード値 %22 で指定します。 ASCII は使用できません。

4. BadURLChars リストから以下の値を削除します。

   ,%25

5. LocalConfig.conf ファイルへの変更を保存し、テキスト エディタを閉じます。
6. 変更するすべての Web サーバ上で手順 1 ～ 5 を繰り返します。

   Windows 認証情報コレクタが TARGET URL 内の HTTP エンコード文字を処理できるようになりました。

FCC のパフォーマンスの調整

認証情報コレクタのパフォーマンスを改善するために、以下の任意の設定が可能です。

- FCC レルム コンテキスト確認の無効化によるパフォーマンスの向上
- フォーム キャッシュの使用

FCC レルム コンテキスト確認の無効化によるパフォーマンスの向上

フォーム認証時に、Web エージェントはポリシー サーバに対し IsProtected 呼び出しを行って、要求されたリソースが保護されているかどうかを判断します。この最初の呼び出しの後、Web エージェントは通常、ポリシー サーバに対する追加の IsProtected 呼び出しを行います。 この 2 番目の呼び出しによってレルム コンテキストが確立され、Web エージェントは FCC を使用したユーザのログインを許可し、保護されているリソースにアクセスできるようにします。 Web エージェントがこの追加呼び出しを行うかどうかは、以下のパラメータを使用して制御することができます。

FCCForceIsProtected

Web エージェントはポリシー サーバに対しもう一度 IsProtected 呼び出しを行うかどうかを指定します。この 2 回目の呼び出しによってレルム コンテキストが確立され、Web エージェントはユーザのログインを許可し、保護されているリソースにアクセスできるようにします。

このパラメータを no に設定すると、Web エージェントはポリシー サーバへの IsProtected の最初の呼び出しから取得されたレルム情報を代わりに使用します。

デフォルト： yes

FCC レルム コンテキスト確認を無効にすることによってパフォーマンスを向上させるには、FCCForceIsProtected パラメータの値を no に設定します。

フォーム キャッシュ

フォーム キャッシュはフォーム テンプレート データを格納します。テンプレート データを格納することにより、エージェントが同じデータに対して何回も .fcc ファイルを読み取らなくなるため、パフォーマンスが改善されます。 FCC の拡張子を持つリソースへのアクセスが発生すると、FCC は対応するテンプレート ファイルを読み取り、処理します。 エージェントはこのような読み取り動作を 1 秒間に数百回実行します。

フォーム キャッシュは、簡単に読み取ることができるメモリにフォーム テンプレート ファイルを保管することで FCC の処理を支援します。 仮想メモリ アクセスはディスク アクセスよりも高速なので、FCC コンポーネントはフォームを短時間で処理でき、ホスト サーバにかかる負荷が軽減されます。

処理時間の短縮により、FCC が各 Web サーバの要求に応える能力が向上します。 フォーム認証は効率化されます。

フォーム キャッシュ データ

フォーム キャッシュに保管されるデータはフォーム テンプレート テキストから成り立ちます。このテキストはデータ構造に事前に解析済みです。このデータ構造により FCC の処理が最適化されます。

データ構造に含まれるものは、以下のとおりです。

- 国際化用のフォーム ロケール データ
- UTF-8 フォーマットの未処理のテキスト、テンプレート ディレクティブ情報、および要求環境からの置換用の関数/変数情報を含むデータ オブジェクトの順序付きリスト

ディレクティブ、関数、および変数は、FCC ファイルの上から下に順に処理されます。

フォーム キャッシュの設定

フォームは、パフォーマンスを改善し、不要なネットワーク トラフィックを削減するためにキャッシュできます。 以下のパラメータを使用して、フォーム キャッシュの設定を制御できます。

EnableFormCache

フォーム テンプレート キャッシュを制御します。このパラメータを yes に設定すると、フォーム認証のパフォーマンスが向上します。キャッシュを無効にするには、このパラメータを no に設定します。

デフォルト： yes

FormCacheTimeout

オブジェクトをキャッシュに保管しておくことのできる期間を秒数で指定します。この期間を過ぎると、そのオブジェクトは無効と見なされます。 タイムアウト間隔を過ぎると、フォーム テンプレート ファイルの日時と、キャッシュ オブジェクトが作成された時刻が比較されます。 キャッシュに保管されたオブジェクトがディスク上のファイルより新しい場合、タイムアウトはリセットされて次のタイムアウトまでオブジェクトは保管されます。 それ以外の場合、オブジェクトはキャッシュから削除されます。

デフォルト： 600

次の手順に従ってください:

1. EnableFormCache パラメータの値を yes に設定します。
2. フォーム キャッシュのタイムアウト間隔を変更する場合は、FormCacheTimeout の値を希望の秒数に設定します。

フォーム キャッシュが設定されます。

NTLM 認証情報コレクタの指定

NTLM 認証情報コレクタ（NTC）は、Web エージェント内のアプリケーションです。 NTC は、Windows 認証方式によって保護されるリソースに関連する NT 認証情報を収集します。 この方式は、Internet Explorer ブラウザからアクセスされる、IIS Web サーバ上のリソースに適用できます。

各認証情報コレクタには、1 つの MIME タイプが関連付けられています。IIS に関しては、以下のパラメータで NTC MIME TYPE が定義されています。

NTCExt

NTLM 認証情報コレクタと関連付けられた MIME タイプを指定します。 このコレクタは、Windows 認証方式によって保護されているリソースに関連する NT 認証情報を収集します。 この方式は、Internet Explorer ブラウザのユーザのみがアクセスする IIS Web サーバ上のリソースに適用できます。

このパラメータに複数の拡張子を持たせることもできます。 エージェント設定オブジェクトを使用している場合は、複数値オプションを選択します。 ローカル設定ファイルを使用している場合は、各拡張子をカンマで区切ります。

デフォルト： .ntc

使用している環境で NTCExt パラメータによって指定されたデフォルトの拡張子がすでに使用されている場合は、別の MIME タイプを指定できます。

認証情報コレクタをトリガする拡張子を変更するには、他のファイル拡張子を NTCExt パラメータに追加します。

4.x タイプと、より新しいタイプのエージェント間での認証情報コレクタの使用

SiteMinder エージェント オブジェクトの古いバージョンでは、ポリシー サーバおよび WebAgent.conf ファイルに格納した共有秘密キーを特徴とするセキュリティ モデルを使用していました。 これらのエージェントを 4.x タイプ エージェントと言います。SiteMinder 管理 UI でエージェント オブジェクトを作成する際に、ユーザは 4.x エージェント機能のサポートを指定できます。

SiteMinder のより新しいバージョンでは、共有秘密キーセキュリティ モデルの代わりにポリシー サーバ上のトラステッド ホスト オブジェクトを使用します。

SiteMinder は 4.x タイプとより新しいエージェント間の認証情報コレクタの使用をサポートしています。 こうした認証情報コレクタの使用方法を混在モードと呼びます。 混在モードを展開するには追加の設定手順が必要です。

混在環境での認証情報コレクタの設定

SiteMinder r6.x から SiteMinder 12.52 では、認証情報コレクタは古い 4.x タイプの認証情報コレクタとは異なる動作をします。 4.x タイプの認証情報コレクタはユーザのブラウザに Cookie を配置し、次にユーザを元のエージェントに再度リダイレクトします。

それ以降の SiteMinder バージョンでは、認証情報コレクタは要求されたリソースを保護するエージェントのために、ポリシー サーバにユーザをログインさせます。 Cookie は使用 *されません*。

注: Cookie を設定するのではなく、認証情報コレクタを使用して直接ユーザをログインさせることを推奨します。 認証情報コレクタを使用したユーザのログインの方が、ユーザの認証情報のセキュリティ保護が強化されます。これは、これらの認証情報が Cookie に格納された状態でネットワーク上を転送されないためです。

認証情報コレクタは、ユーザをログインさせるために次の情報が必要です。

- 要求されたリソースを保護しているエージェントの名前。
- ユーザによって提供される認証情報。

エージェント名を把握するために、認証情報コレクタは以下のプロセスを使用します。

1. SMAGENTNAME クエリ パラメータを使用します。これを、元のエージェントは認証情報コレクタへリダイレクトする際に URL のクエリ文字列に追加します。
2. エージェント名が URL に追加されない場合は、認証情報コレクタと関連付けられた AgentName 設定パラメータで定義されたマッピングを使用します。

   AgentName パラメータの各マッピングは、コレクタを使用して自らのリソースを保護しているホストの名前と、その IP アドレスを指定します。
3. エージェント名マッピングが設定されていない場合は、エージェント名として、ターゲット URL の完全修飾ホスト名を使用します。この動作は AgentNamesAreFQHostNames 設定パラメータを有効にすることにより決定されます。

   このパラメータは、デフォルトで無効になっています。その場合、認証情報コレクタはエージェント名として、DefaultAgentName パラメータの値を使用します。

混在環境で認証情報コレクタを設定する前に、前出の関係を考慮してください。

混在環境での FCC と NTC の使用

リクエストを処理する上で、FCC と NTC はユーザ認証情報、およびリクエストされたリソースを保護している Web エージェントの名前を必要とします。 ただし、4.x エージェント、および FCC および NTC へのポストを行うサードパーティのエージェントは、自らが送信する URL の一部としてエージェント名を渡すことはありません。

FCC と NTC を 4.x の Web エージェントと組み合わせて使用する場合は、以下の設定オプションが役立ちます。

互換モードを使用する - 4.x エージェントまたはサードパーティのアプリケーションによって保護されているリソースに対してフォームを提供するように r5.x、r6.x、または 12.52 の FCC/NTC を有効にしてから、FCCCompatMode パラメータを有効にします。従来の Web エージェントでは FCCCompatMode パラメータはデフォルトで有効です。 フレームワークエージェントでは、FCCCompatMode パラメータはデフォルトで無効です。

このパラメータを有効にすると、r5.x、r6.x、または 12.52 エージェントは 4.x エージェントのようにフォームおよび NTLM 認証情報コレクションを操作できるようになります。この設定（フォームまたは NTLM 認証情報 Cookie がユーザのブラウザに書き込まれることを意味します）は、ログインの前にエージェントに再度リダイレクトされます。この設定で、エージェントの相互運用が可能になります。

FCCCompatMode パラメータの値が No に設定されている場合、4.x のエージェントとの互換性は無効になります。12.52 環境では、パラメータの値を No に設定してください。

重要: このパラメータを no に設定すると、Netscape ブラウザのバージョン 4.x のサポートが削除されます。

- エージェント名のマッピングを指定する - FCC のみ。下位互換性を無効にした場合は、AgentName パラメータを、FCC を使用してリソースを保護している各ホストの名前と IP にマップします。これらのマッピングは FCC の設定でセットアップします。

  マッピングの例 :

  myagent, 123.1.12.1

  myagent, www.sitea.com

- ホスト名をエージェント名として使用する - FCC のみ。アルゴリズム内の最初の 2 つのオプションが適していない場合は、AgentNamesAreFQHostNames パラメータの値を Yes に設定することができます。 これは、ターゲット URL の中にある完全修飾ホスト名をエージェント名として使用するよう FCC に指示します。たとえば、URL 文字列の中に、次の内容が含まれているとします。

  url?A=1&Target=http://www.nete.com/index.html

  Target 文字列のうち、www.nete.com の部分がエージェント名として提供されます。

  デフォルトでは、このパラメータは no に設定されます。その結果、DefaultAgentName パラメータの値がエージェント名として使用されます。

以下の表に、r5.x、r6.x、または 12.52 と 4.x の FCC および NTC を設定する場合のガイドラインを示し、混在環境でのそれぞれがどのように動作するかについて説明します。

注:

- NTLM 認証情報コレクタは、IIS 以外の Web サーバから IIS Web サーバへユーザをリダイレクトすることができます。
- フレームワーク Web エージェントに関しては、FCC 互換モードが無効になっている状態に関する説明のみを参照してください。

リソースを保護する Web エージェント	FCC 互換モードでの r5.x、r6.x、または 12.52 の FCC	r5.x、r6.x、または 12.52 の FCC - FCC 互換モード無効
r5.x、r6.x、または 12.52	■ FCC は認証情報 Cookie を発行します。 ■ 「証明書**および**フォームによる認証」は無効です。 ■ 「証明書**または**フォームによる認証」は無効です。	■ FCC は、セッション Cookie を発行します。 ■ 「証明書**および**フォームの組み合わせによる認証」は機能します。 ■ 「証明書**または**フォームによる認証」は機能します。

リソースを保護する Web エージェント	4.x QMR 2/3/4 の FCC
4.x QMR 5 または 4.x QMR 6	■ エージェントは認証情報 Cookie を発行します。 ■ 「証明書およびフォームによる認証」は無効です。 ■ 「証明書またはフォームによる認証」は機能します。
r5.x、r6.x、または 12.52	■ エージェントは認証情報 Cookie を発行します。 ■ 「証明書およびフォームによる認証」は無効です。 ■ 「証明書またはフォームによる認証」は機能します。

注: SSL 認証方式の詳細については、ポリシー サーバのマニュアルを参照してください。

リソースを保護する Web エージェント	FCC 互換モードでの r5.x、r6.x、または 12.52 の FCC	r5.x、r6.x、または 12.52 の FCC - FCC 互換モード無効
4.x QMR 5 または 4.x QMR 6	■ NTC は認証情報 Cookie を発行します。	■ NTC は、セッション Cookie を発行します。
r5.x、r6.x、または 12.52	■ NTC は認証情報 Cookie を発行します。	■ NTC は、セッション Cookie を発行します。

リソースを保護する Web エージェント	4.x QMR 2/3/4 の NTC
4.x QMR 5、または 4.x QMR 6	■ エージェントは認証情報 Cookie を発行します。
r5.x、r6.x、または 12.52	■ エージェントは認証情報 Cookie を発行します。

混在環境での SCC の使用

4.x タイプの Web エージェントと r5.x、r6.x、または 12.52 の SCC の相互運用を可能にするには、以下のいずれかのタスクを実行します。

- エージェント名のマッピングを指定する： AgentName パラメータを、SCC を使用してリソースを保護している各ホストの名前とその IP アドレスにマップします。 SCC のエージェント設定パラメータでこれらのマッピングを作成します。
- ホスト名をエージェント名として使用する：エージェント名のマッピングを指定しない場合は、AgentNamesAreFQHostNames パラメータを Yes に設定することができます。 これは、ターゲット URL の中にある完全修飾ホスト名をエージェント名として使用するよう SCC に指示します。

  たとえば、次のような URL 文字列が発生したとします。

  ```
  url?A=1&Target=http://www.nete.com/index.html
  ```

  Target 文字列のうち、www.nete.com の部分がエージェント名として提供されます。

  デフォルトでは、このパラメータは no に設定されます。その結果、DefaultAgentName パラメータの値がエージェント名として使用されます。

以下の表に、混在環境で SCC として機能する 4.x のエージェントと r5.x、r6.x、または 12.52 のエージェントがどのように動作するかを示します。

Web エージェントのバージョン	4.x QMR 2/3/4 の SCC	r5.x、r6.x、または 12.52 の SCC
4.x QMR 5 または 4.x QMR 6	■ エージェントは SSL 認証情報 Cookie を発行します。 ■ ブラウザからサーバに対する元の接続が SSL 経由であっても、リクエストをリダイレクトしない限り、証明書を収集することはできません。	■ AgentName パラメータでマッピングを作成するか、AgentNamesAreFQHostNames を Yes に設定します。 ■ SCC は、セッション Cookie を発行します。 ■ ブラウザからサーバに対する元の接続が SSL 経由であっても、リクエストをリダイレクトしない限り、証明書を収集することはできません。

Web エージェントのバージョン	4.x QMR 2/3/4 の SCC	r5.x、r6.x、または 12.52 の SCC
r5.x、r6.x、または 12.52	■ エージェントは SSL 認証情報 Cookie を発行します。 ■ リクエストをリダイレクトすることなく、証明書を収集できます。	■ SCC は、セッション Cookie を発行します。 ■ リクエストをリダイレクトすることなく、証明書を収集できます。

注: SSL 認証方式の詳細については、ポリシー サーバのマニュアルを参照してください。

日本語環境の FCC ベースのパスワード サービスに対する Apache ベース エージェントの設定

日本語のオペレーティング環境で使用される smpwservices.fcc ファイルには、不正なコード設定があります。 この不正な設定により、Web ページに不正な文字が表示される可能性があります。 ディレクティブを Apache ベースの Web サーバの httpd.conf ファイルに追加することにより、この問題を修正します。

次の手順に従ってください:

1. Apache ベースの Web サーバにログインします。
2. httpd.conf ファイルをテキスト エディタで開きます。
3. ファイルの最後に空白行を追加します。
4. 以下のディレクティブを空白行に追加します。

   ```
   BrowserMatch ".*" suppress-error-charset
   ```

5. httpd.conf ファイルを保存し、テキスト エディタを閉じます。
6. Apache ベースの Web サーバを停止します。
7. Apache ベースの Web サーバを起動します。

第 12 章: FCC 国際化

このセクションには、以下のトピックが含まれています。

FCC の国際化を有効にする方法

SiteMinder は、FCC の国際化に対応するために以下の機能をサポートします。

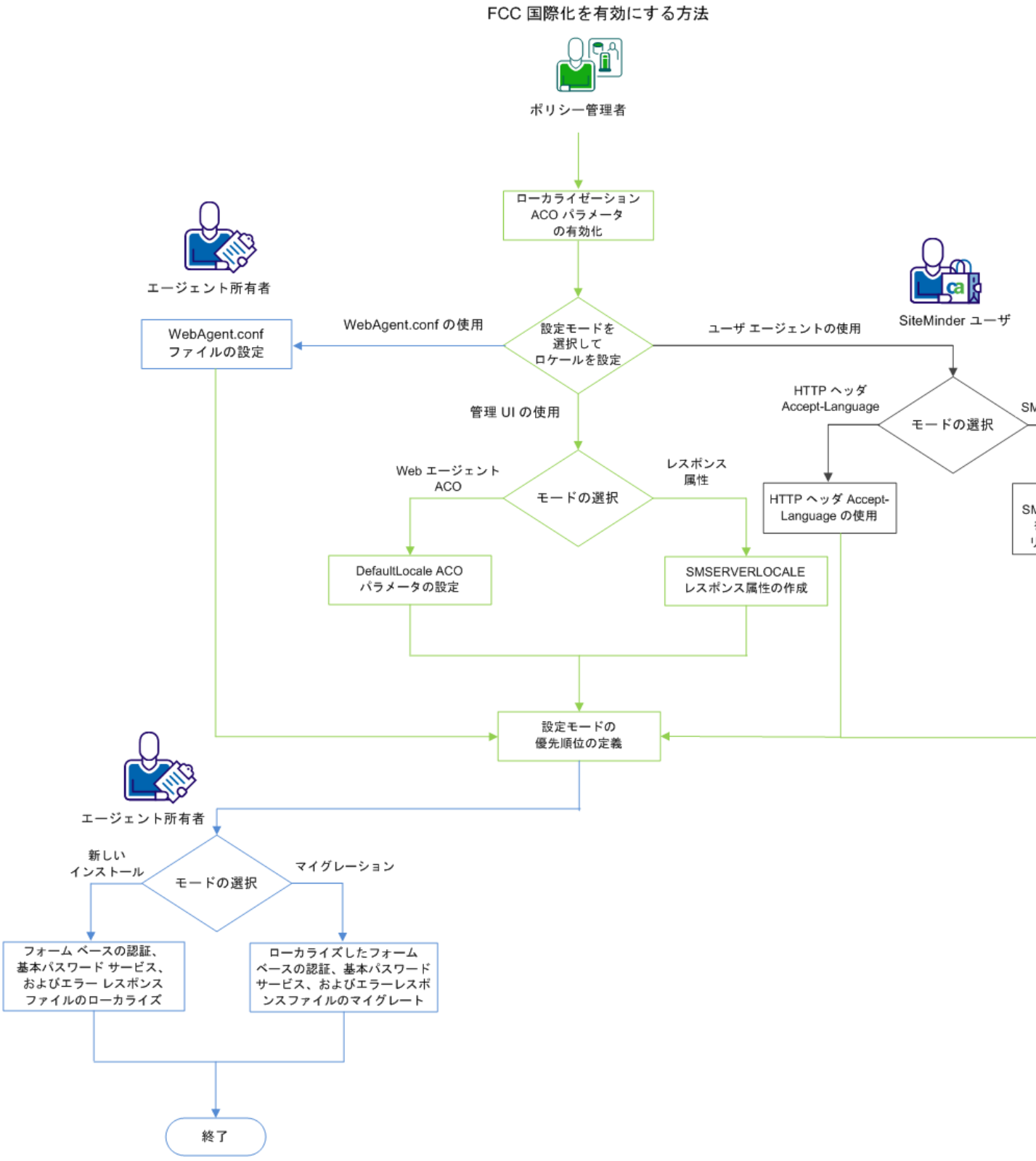
- 認証方式に対する FCC
- 基本的なパスワード サービス
- エラー レスポンス ページ

SiteMinder がログイン、基本的なパスワード サービスおよびエラー レスポンス用に HTML ページを表示する必要がある優先ロケールを設定できます。 ユーザの組織にこのシナリオを実装するには、以下のユーザ ロールが必要です。

- ポリシー管理者
- エージェント所有者
- SiteMinder ユーザ

以下の図は、この機能をサポートするために各ユーザが実行する必要がある手順について説明しています。

注: 以下の図で、青色は、ポリシー管理者のタスクを示します。緑色は、エージェント所有者のタスクを示します。黒色は、SiteMinder ユーザのタスクを示します。


FCC 国際化を有効にする方法
ポリシー管理者
ローカライゼーション
ACO パラメータ
の有効化
エージェント所有者
SiteMinder ユーザ
WebAgent.conf
ファイルの設定
WebAgent.conf の使用
設定モードを
選択して
ロケールを設定
ユーザ エージェントの使用
HTTP ヘッダ
Accept-Language
モードの選択
管理 UI の使用
Web エージェント
ACO
モードの選択
レスポンス
属性
HTTP ヘッダ Accept-
Language の使用
DefaultLocale ACO
パラメータの設定
SMSERVERLOCALE
レスポンス属性の作成
設定モードの
優先順位の定義
エージェント所有者
新しい
インストール
モードの選択
マイグレーション
フォーム ベースの認証、
基本パスワード サービス、
およびエラー レスポンス
ファイルのローカライズ
ローカライズしたフォーム
ベースの認証、基本パスワード
サービス、およびエラーレスポ
ンスファイルのマイグレート
終了

FCC の国際化を有効にするには、以下の手順を実行する必要があります。

1. 管理 UI を使用して、Localization ACO パラメータを有効にします (P. 243)。 ポリシー管理者は、このタスクを実行する必要があります。
2. 優先ロケールを設定するために、設定モードを選択します (P. 244)。このタスクは、以下の方法で実行できます。
   - ユーザ エージェントの使用。SiteMinder ユーザは、ユーザ エージェントを使用して、以下の方法で優先ロケールを設定できます。
     - HTTP ヘッダ SMCLIENTLOCALE を使用します。
     - HTTP ヘッダ Accept-Language (P. 244) を使用します。
   - 管理 UI の使用 (P. 245)。 ポリシー管理者は、管理 UI を使用して、以下の方法で優先ロケールを設定できます。
     - SMSERVERLOCALE レスポンス属性を使用します (P. 245)。
     - DefaultLocale ACO パラメータを使用します (P. 246)。
   - the WebAgent.conf ファイルの使用 (P. 246)。エージェント所有者は、WebAgent.conf ファイル内の Locale パラメータを設定して、優先ロケールを設定できます。
3. 設定モードの優先順位を定義します (P. 247)。 ポリシー管理者は管理 UI を使用して、このタスクを実行する必要があります。
4. 以下の手順のいずれかを実行します。
   - HTML ページを初めてローカライズする場合、エージェント所有者は以下の手順を実行する必要があります。
     a. フォーム ベースの認証ファイルおよび基本的なパスワード サービス ファイルをローカライズします (P. 249)。
     b. エラー レスポンス ファイルをローカライズします (P. 251)。
   - ローカライズされたファイルを移行する場合、エージェント所有者は以下の手順を実行する必要があります。
     a. ローカライズされたフォーム ベースの認証ファイルおよび基本的なパスワード サービス ファイルを移行します (P. 256)。
     b. ローカライズされたエラー レスポンス ファイルを移行します (P. 258)。

   SiteMinder は、FCC の国際化を有効にするように設定されます。

ローカライゼーション パラメータの有効化

ポリシー管理者は、Localization ACO パラメータの値を true に設定して、サポートされている機能をローカライズする必要があります。

次の手順に従ってください:

1. 管理 UI にログオンします。
2. ［インフラストラクチャ］-［エージェント］をクリックします。
3. ［エージェント設定オブジェクト］をクリックします。

   ［エージェント設定オブジェクト］ページが表示されます。
4. Web エージェントのエージェント設定オブジェクトに移動します。
5. エージェント設定オブジェクトの名前をクリックします。

   ［エージェント設定の表示］ページが表示されます。
6. ［変更］をクリックします。

   設定とコントロールがアクティブになります。
7. Localization パラメータに移動し、パラメータの左側の右方向矢印をクリックします。

   ［パラメータの編集］ページが表示されます。
8. パラメータの値を yes に設定します。
9. 変更を保存します。

優先ロケールの設定

SiteMinder では、Accept-Language HTTP ヘッダ形式で、複数の優先ロケールを定義できます。

例： `es-ES,en-US;q=0.8,as-IN;q=0.6,fr-FR;q=0.4,en-IN;q=0.2`

SiteMinder は、実行時に動的にロケールを決定しますが、ロケールはリクエスト処理中に変更できます。 たとえば、ロケールは認証成功後に変わる場合があります。

以下の設定モードで、優先ロケールを定義します。

- ユーザ エージェントの使用
- 管理 UI の使用
- WebAgent.conf ファイルの使用

ユーザ エージェントを介した優先ロケールの設定

SiteMinder ユーザは、Web ブラウザ設定を設定して、SiteMinder がサポートされている機能に関連するリクエストに対応する必要がある優先ロケールを設定できます。 SiteMinder ユーザは、以下の方法を使用してロケールを設定できます。

- HTTP ヘッダ SMCLIENTLOCALE の使用
- HTTP ヘッダ Accept-Language の使用

HTTP ヘッダ SMCLIENTLOCALE の使用

SiteMinder ユーザは、Web リクエストの HTTP ヘッダ SMCLIENTLOCALE を使用して、優先ロケールを設定できます。 Web アプリケーションが優先ロケールを設定するオプションを提供する場合、SiteMinder ユーザはロケールを選択できます。SiteMinder は Web リクエストで SMCLIENTLOCALE ヘッダを使用して、ロケールを受信します。 ロケールが利用可能な場合、SiteMinder はリクエストされたロケールで、サポートされている機能のページを表示します。

HTTP ヘッダ Accept-Language の使用

SiteMinder はローカル Web ブラウザ設定を使用して、優先ロケールを設定できます。 Web リクエストの HTTP ヘッダ Accept-Language はロケールを指定します。

管理 UI を介した優先ロケールの設定

ポリシー管理者は管理 UI を使用して、SiteMinder がサポートされている機能に関連するリクエストに対応する必要がある優先ロケールを設定できます。

SMSERVERLOCALE レスポンス ヘッダ変数の設定

ポリシー管理者は、 WebAgent-HTTP-Header 変数属性タイプをベースとした SMSERVERLOCALE レスポンス属性を使用して、SiteMinder レスポンスとして優先ロケールを設定します。

次の手順に従ってください:

1. 管理 UI にログオンします。
2. ［ポリシー］-［ドメイン］をクリックします。
3. ［レスポンス］をクリックします。

   ［レスポンス］ページが表示されます。
4. 変更するレスポンスを編集します。
5. ［レスポンスの定義］タブをクリックします。
6. 変更するレスポンス属性を編集します。

   ［レスポンス属性の表示］ページが表示されます。
7. WebAgent-HTTP-Header 変数として［属性タイプ］を選択します。
8. ［変数名］に「SMSERVERLOCALE」と入力します。
9. ［変数値］に優先ロケールを入力します。
10. 変更内容を保存します。

DefaultLocale ACO プロパティの設定

ポリシー管理者は、DefaultLocale ACO パラメータで優先ロケール情報を設定できます。

次の手順に従ってください:

1. 管理 UI にログオンします。
2. ［インフラストラクチャ］-［エージェント］をクリックします。
3. ［エージェント設定オブジェクト］をクリックします。

   ［エージェント設定オブジェクト］ページが表示されます。
4. Web エージェントのエージェント設定オブジェクトに移動します。
5. エージェント設定オブジェクトの名前をクリックします。

   ［エージェント設定の表示］ページが表示されます。
6. ［変更］をクリックします。

   設定とコントロールがアクティブになります。
7. DefaultLocale パラメータに移動し、パラメータの左側の右方向矢印をクリックします。

   ［パラメータの編集］ページが表示されます。
8. 優先ロケールで、プロパティを更新します。
9. 変更を保存します。

WebAgent.conf ファイルを使用した優先ロケールの設定

エージェント所有者は、WebAgent.conf ファイルを設定して、Locale プロパティを設定する必要があります。このプロパティには、インストール中に指定されるロケール情報が含まれます。

次の手順に従ってください:

1. Web エージェント マシンにログオンします。
2. テキスト エディタで、Web エージェントの WebAgent.conf ファイルを開きます。
3. Locale プロパティに移動し、値を優先ロケールに設定します。
4. 変更内容を保存します。

設定モードの優先順序の定義

ポリシー管理者は、ClientLocalePreferred ACO パラメータを設定して、5 つの設定モードの優先順位を決定する必要があります。 この値が true に設定される場合、SiteMinder は以下の優先順位でロケールを確定するために、設定モードを確認します。

1. HTTP ヘッダ SMCLIENTLOCALE
2. HTTP ヘッダ Accept-Language
3. レスポンス ヘッダ変数 SMSERVERLOCALE
4. DefaultLocale ACO パラメータ
5. WebAgent.conf ファイル内の Locale プロパティ

この値が false に設定される場合、SiteMinder は以下の優先順位でロケールを確定するために、設定モードを確認します。

1. レスポンス ヘッダ変数 SMSERVERLOCALE
2. HTTP ヘッダ SMCLIENTLOCALE
3. HTTP ヘッダ Accept-Language
4. DefaultLocale ACO パラメータ
5. WebAgent.conf ファイル内の Locale プロパティ

SiteMinder は、5 つの設定モードを使用して定義された優先ロケールを処理し、各優先ロケールの q ファクタをベースに並べ替えられた、優先ロケールのリストを作成します。 Web リクエストが送信される場合、SiteMinder は優先ロケールのリストを使用して、サポートされている機能の HTML ページが表示される必要があるロケールを確定します。

次の手順に従ってください:

1. ［インフラストラクチャ］-［エージェント］をクリックします。
2. ［エージェント設定オブジェクト］をクリックします。

   ［エージェント設定オブジェクト］ページが表示されます。
3. Web エージェントのエージェント設定オブジェクトに移動します。
4. エージェント設定オブジェクトの名前をクリックします。

   ［エージェント設定の表示］ページが表示されます。
5. ［変更］をクリックします。

6. 設定とコントロールがアクティブになります。
7. ClientLocalePreferred パラメータに移動します。
   ［パラメータの編集］ページが表示されます。
8. パラメータ値を変更します。
9. 変更を保存します。

フォーム ベースの認証および基本的なパスワード サービス ファイルのローカライズ

エージェント所有者は、SiteMinder によって表示されているフォーム ベースの認証および基本的なパスワード サービス ファイルをローカライズできます。 デフォルトでは、SiteMinder は Web エージェントの以下の場所で、さまざまなロケールのサンプル フォームを提供します。

webagent_home/samples

各ロケールには、ロケール固有のファイル用のフォルダがあります。 各フォルダおよびファイルの末尾に、命名規則の RFC 3066 標準に準拠するロケール言語コードが、以下の形式で追加されます。

forms_<*ロケール言語コード*>

forms_<*ロケール言語コード*>/*filename*_<*ロケール言語コード*>.fcc

例：

日本語の場合、フォルダ名は samples/forms_ja-JP、login.fcc ファイル名は login_ja-JP.fcc、WebAgent.properties ファイル名は WebAGent_ja-JP.properties です。

サンプル フォーム フォルダに加えて、SiteMinder では、ロケール言語コードのないファイルが含まれるデフォルト フォーム フォルダも提供されます。 Web エージェントがロケールのリクエストを受け取ると、SiteMinder は優先ロケール リスト内のロケールを処理して、リクエストされたロケールが利用可能かどうか確認し、以下のタスクの*いずれか*を実行します。

- リクエストされたロケールが優先ロケール リスト内のロケールに一致する場合、SiteMinder はリクエストされたロケールでリクエストを処理します。
- リクエストされたロケールが優先ロケール リストに一致しない場合、SiteMinder はデフォルト フォーム フォルダのロケールを使用して、リクエストを処理します。

新規ロケールをサポートするための SiteMinder のカスタマイズ

デフォルトでは、デフォルト フォーム フォルダは en-US にあります。エージェント所有者は、デフォルトのフォルダおよびファイルをカスタマイズして、SiteMinder によってデフォルトでは提供されない新規ロケールをサポートする必要があります。

次の手順に従ってください:

1. Web エージェント マシンにログオンします。
2. 以下の場所に移動します。

   webagent_home/samples
3. 以下の形式で、新規ロケール用のフォルダを作成します。

   forms_<*ロケール言語コード*>
4. .fcc ファイルをカスタマイズするには、以下の手順を実行します。
   a. 関連する FCC のファイルを、デフォルトのフォーム フォルダから新規フォルダへ配置します。
   b. 各 FCC ファイルの末尾に、ロケールの言語コードを以下の形式で追加します。

      *filename*_<*ロケール言語コード*>.fcc
5. 以下の手順を実行して、.fcc ファイルをカスタマイズします。
   a. 関連する .properties ファイルを、デフォルトのフォーム フォルダからユーザのロケールのフォーム フォルダに配置します。
   b. 各ファイルの末尾に、ユーザのロケールの言語コードを以下の形式で追加します。

      *filename*_<*ロケール言語コード*>.properties
6. 英語のメッセージのユーザのロケール言語への変更など、ロケールに対応するようにファイルを変更します。
7. 変更内容を保存します。

エラー レスポンス ファイルのローカライズ

エージェント所有者は、SiteMinder で表示されるエラー レスポンス ファイルをローカライズできます。 デフォルトでは、SiteMinder はさまざまなロケールのサンプル エラー レスポンスを以下の場所で提供します。

webagent_home/samples/error-responses

各ロケールには、ロケール固有のファイル用のフォルダがあります。 各フォルダおよびファイルの末尾に、命名規則の RFC 3066 標準に準拠するロケール言語コードが、以下の形式で追加されます。

responses_<*ロケール言語コード*>
filename_<*ロケール言語コード*>.err

サンプル エラー レスポンス フォルダに加えて、SiteMinder は、ロケール言語コードが含まれないデフォルト エラー レスポンス フォルダおよびファイルも提供します。 Web エージェントがロケールでリクエストを受け取ると、SiteMinder は以下のタスクのいずれかを実行します。

- ロケールが利用可能で、エラーが発生する場合、SiteMinder は対応するロケール フォルダにあるエラー レスポンス ファイルを使用して、エラー レスポンスを表示します。
- ロケールが利用可能でなく、エラーが発生する場合、SiteMinder はデフォルト エラー レスポンス フォルダにあるファイルを使用して、エラー レスポンスを表示します。

新規ロケールをサポートするための SiteMinder のカスタマイズ

デフォルトのフォルダおよびファイルをカスタマイズして、SiteMinder によってデフォルトでは提供されない新規ロケールをサポートします。

以下のエラー レスポンス ファイルは、ユーザの優先ロケールに適するようにカスタマイズできます。

- CSS エラーに対して生成される csserror.err
- サーバ エラーに対して生成される servererror.err
- Cookie エラーに対して生成される cookieerror.err
- 認証および許可エラーに対して生成される custom401error.err

エラーが発生する場合、エラー ページは以下の場所から取得されます。

ErrorResponseLocation_home/responses_<*ロケール言語コード*>

例：

SiteMinder は、日本語ロケールのエラー レスポンス ファイルを *ErrorResponseLocation_home* /responses_ja-JP フォルダに格納します。Cookie エラーが発生する場合、cookierror_ja-JP.err エラー レスポンス ファイルは *ErrorResponseLocation_home*/responses_ja-JP フォルダから取得されます。

エラー レスポンスをカスタマイズするには、以下の手順を実行します。

1. Web エージェントを設定する。
2. Web エージェント ACO を設定する。

Web エージェントの設定

エージェント所有者は、Web エージェントを設定してエラー レスポンス ファイルをローカライズする必要があります。

次の手順に従ってください:

1. Web エージェント マシンにログオンします。
2. 以下の場所に移動します。

   webagent_home/samples/error-responses
3. 以下の形式で、新規ロケール用のフォルダを作成します。

   responses_<ロケール言語コード>
4. デフォルトのレスポンス フォルダから新しいフォルダへ、レスポンス ファイルを配置します。
5. 各エラー レスポンス ファイルの末尾に、ユーザのロケールの言語コードを以下の形式で追加します。

   filename_<ロケール言語コード>
6. ロケールに合わせてファイルを変更します。たとえば、英語のメッセージをロケールの言語に変更します。

Web エージェント ACO の設定

ポリシー管理者は、Web エージェントの ACO パラメータを設定して、エラー レスポンス ファイルをローカライズする必要があります。

次の手順に従ってください:

1. 管理 UI にログオンします。
2. ［インフラストラクチャ］-［エージェント］をクリックします。
3. ［エージェント設定オブジェクト］をクリックします。

   ［エージェント設定オブジェクト］ページが表示されます。
4. Web エージェントのエージェント設定オブジェクトに移動します。
5. エージェント設定オブジェクトの名前をクリックします。

   ［エージェント設定の表示］ページが表示されます。
6. ［変更］をクリックします。

   設定とコントロールがアクティブになります。
7. 以下のパラメータの値が no に設定されていることを確認します。
   - servererrorfile
   - csserrorfile
   - reqcookieerrorfile
   - custom401errorfile

   注: これらのプロパティの優先度は ErrorResponseLocation パラメータより上です。 エラー レスポンス ファイルをローカライズするには、これらのプロパティが無効であることを確認します。
8. ErrorResponseLocation パラメータに移動し、パラメータの左側の右方向矢印をクリックします。

   ［パラメータの編集］ページが表示されます。
9. エラー レスポンス ファイルの新しいパスを入力します。

   注: ErrorResponseLocation プロパティの値は URL にはできません。
10. 変更内容を保存します。

ローカライズされたファイルの移行

SiteMinder 12.51 より前のリリースで FCC、ベース パスワード サービスおよびエラー レスポンスのローカライズされたファイルが利用可能な場合、エージェント所有者は、それらを SiteMinder 12.51 以降のリリースに移行して、再利用できます。

フォーム ベースの認証ファイルおよび基本的なパスワード サービス ファイルの移行

エージェント所有者はフォーム ベースの認証ファイルおよび基本的なパスワード サービス ファイルを移行できます。

次の手順に従ってください:

1. Web エージェント マシンにログオンします。
2. ローカライズされたファイルが存在する場所に移動します。

   デフォルト パス： *webagent_home*/samples
3. 以下の手順のいずれかを実行します。
   - この場所にフォルダを作成する場合は、以下の手順を実行します。
     a. 以下の形式でフォルダを作成します。

     `forms_<`*ロケール言語コード*`>`

     b. 既存のローカライズされた FCC ファイルを新しいフォルダに配置します。

     c. 各ファイルの末尾に、ユーザのロケールの言語コードを以下の形式で追加します。

     `filename_<`*ロケール言語コード*`>.fcc`

     d. 関連する .properties ファイルを、既存のフォルダからユーザのロケールのフォーム フォルダに配置します。

     e. 各ファイルの末尾に、ユーザのロケールの言語コードを以下の形式で追加します。

     `filename_<`ロケール言語コード`>.properties`
   - この場所の既存のフォルダの名前を変更する場合は、以下の手順を実行します。
     a. 以下の形式で、既存フォルダの名前を変更します。

     `forms_<`*ロケール言語コード*`>`

     b. 各ファイルの末尾に、ユーザのロケールの言語コードを以下の形式で追加します。

     `filename_<`*ロケール言語コード*`>.fcc`

     c. 各ファイルの末尾に、ユーザのロケールの言語コードを以下の形式で追加します。

     `filename_<`ロケール言語コード`>.properties`
4. FCC ファイルで以下の手順を実行します。

a. ハードコードされたエンコーディング パラメータを $$SMENC$$ タグに置換します。

例：

- <meta http-equiv="Content-Type" content="text/html;charset=$$SMENC$$">
- <INPUT TYPE=HIDDEN NAME="SMENC" VALUE="$$SMENC$$">

b. 各ファイルの最初の行には、フォームのエンコードに使用される文字セットが含まれる必要があります。 例： <!-- SiteMinder Encoding=UTF-8; -->。この例では、UTF-8 がフォームのエンコードに使用されます。

5. 変更内容を保存します。

エラー レスポンス ファイルの移行

エージェント所有者は、ローカライズされたエラー レスポンス ファイルを移行できます。

次の手順に従ってください:

1. Web エージェント マシンにログオンします。
2. ローカライズされたファイルが存在する場所に移動します。

   デフォルト パス： *webagent_home*/samples/error-responses
3. フォルダを作成する場合は、以下の手順を実行します。
   a. 以下の形式でフォルダを作成します。

      `responses_<ロケール言語コード>`

   b. ローカライズされた既存のエラー レスポンス ファイルを新しいフォルダに配置します。

   c. 各ファイルの末尾に、ユーザのロケールの言語コードを以下の形式で追加します。

      `filename_<ロケール言語コード>`
4. 既存フォルダの名前を変更する場合は、以下の手順を実行します。
   a. 以下の形式で、既存ファイルの名前を変更します。

      `responses_<ロケール言語コード>`

   b. 各ファイルの末尾に、ユーザのロケールの言語コードを以下の形式で追加します。

      `filename_<ロケール言語コード>`

   SiteMinder は、FCC の国際化を有効にするように設定されます。

第 13 章: エージェントおよびパスワード サービス

このセクションには、以下のトピックが含まれています。

FCC パスワード サービスの設定方法

パスワード サービスを設定するには、次の手順に従ってください :

1. 管理 UI を開きます。
2. SiteMinder 環境でユーザ ディレクトリと関連付けられるパスワード ポリシーを作成します。 ［リダイレクト URL］フィールドで以下のパスを使用します。

   ```
   /siteminderagent/forms/smpwservices.fcc
   ```

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

パスワード サービスの実装

SiteMinder は、パスワード サービスをサポートするためにフォーム認証情報コレクタ（FCC）を使用します。

パスワード サービスは、ユーザが以下のタスクを実行するのに便利です。

- パスワード サービス URL のクエリ文字列の暗号化
- パスワード サービスの複数言語でのサポート
- パスワード サービス ユーザの完全修飾 URL へのリダイレクト
- パスワード サービスによる SecureID 認証のサポート
- ユーザは各自のパスワードを変更できます。 状況に合わせて、次のいずれかの手順を使用します。
  - SecureURLs パラメータが No の場合に、パスワードを変更します (P. 265)。
  - SecureURLs パラメータが Yes の場合に、パスワードを変更します (P. 267)。
  - 基本認証または X.509 証明書認証方式を使用する場合に、パスワードを変更します (P. 269)。

FCC パスワード サービスと URL クエリ暗号化

FCC パスワード サービス アプリケーションでは、URL 上のクエリ データを暗号化することで、エージェントの対話を保護することができます。クエリ データは、FCC パスワード サービスでのみ暗号化することができます。FCC パスワード サービスのファイルには以下のものがあります。

- smpwservices.fcc

  この FCC ファイルは Web エージェントと一緒にインストールされます。このファイルの場所は以下のとおりです。

  web_agent_home/samples/forms

  パスワード サービスの呼び出し時に、パスワード ポリシーが設定されていなかった場合、ポリシー サーバの SiteMinder 管理者は、環境変数 NETE_PWSERVICES_REDIRECT を smpwservices.fcc の相対パスに設定する必要があります。

  パスは以下のとおりです。

  /siteminderagent/forms/smpwservices.fcc

  新しい FCC では、FCC ディレクティブ authreason および username に基づいてパスワード サービスのフォームが表示されます。

- smpwservices.unauth

  このファイルは、パスワード サービス フォームの GET/POST アクション時に発生したエラーを処理します。

  このファイルは、POST 時に要求処理で障害が発生した場合に呼び出されるその他の FCC 無許可ファイルと同じです。 この FCC は、空の TARGET 変数などのエラー状況を処理します。 このエラー報告は、CGI ベースのパスワード サービスと同期させて、FCC POST によって生じるその他の不明なエラーを処理することを目的としています。

- smpwservicesUS-EN.properties

  この properties ファイルは、ユーザにわかりやすいメッセージをパスワード サービス フォームに表示するために smpwservices.fcc で使用されます。

  この properties ファイルにはユーザにわかりやすいメッセージが収められています。管理者は、パスワード サービスのフォームにどのメッセージを表示するかに応じてこのファイルを変更することができます。メッセージのフォーマットは name=value です。

FCC ベースのパスワード サービス変更フォームをローカライズする方法

別のロケール用に FCC ベースのパスワード サービスのユーザ メッセージをローカライズするには、以下の手順に従います。

1. 新規ロケール用の FCC フォルダを Web サーバに作成します。または、すでにロケールに適したフォルダがある場合は、その既存のフォルダを使用します。 フォルダの一般的な命名規則は forms*locale* です。

   注: ユーザの操作環境および使用中の Web サーバのタイプに応じて、表示されるディレクトリおよびファイル名は大文字と小文字が区別される場合があります。

2. 新しいフォルダに、関連するパスワード サービス ファイルをコピーします。

3. ロケールに合わせてファイルを変更します。たとえば、英語のメッセージをロケールの言語に変更します。 そのロケールのすべてのファイルについて、この手順を繰り返します。

4. 管理 UI で、［パスワード ポリシー］の［リダイレクト URL］フィールドの値を変更します。

   たとえば、日本語のユーザに FCC パスワード サービスを使用するには、以下のファイルをフォルダ formsja にコピーします。このフォルダは *web_agent_home*/samples にあります。

   - smpwservices.fcc（*web_agent_home*>/samples/forms 内）
   - smpwservices.unauth（*web_agent_home*>/samples/forms 内）
   - 新しい properties ファイル smpwservicesja.properties

パスワード サービス リダイレクトでの完全修飾 URL の使用

パスワード サービスを使用する場合、ユーザがリダイレクトされる場所の完全修飾ドメイン名（FQDN）を作成するように Web エージェントに指示できます。 以下のパラメータを使用します。

ConstructFullPwsvcUrl

ユーザをリダイレクトする前に、パスワード サービスをホストしているシステムのサーバ名（FQDN）を追加するようにエージェントに指示します。 ポリシー サーバのパスワード ポリシーでこのサーバ名を定義します。

たとえば、このパラメータの値が yes であり、パスワード ポリシーが siteminderagent/forms/smpwservices.fcc を指していると仮定します。 Web エージェントは以下の URL にリダイレクトします。

HTTP://*server_name*.example.com/siteminderagent/forms/smpwservices.fcc

このパラメータの値が no の場合、Web エージェントはパスワード ポリシーで定義された値を使用します。たとえば、パスワード ポリシーがサブディレクトリのみを指している場合、Web エージェントはユーザをそのサブディレクトリにリダイレクトします。

デフォルト： No。

例： No （パスワード ポリシーで定義された /siteminderagent/forms/smpwservices.fcc にリダイレクトします）。

例： Yes （HTTP://*server_name*.example.com をパスワード ポリシーで定義された /siteminderagent/forms/smpwservices.fcc に追加します）。

管理 UI のパスワード ポリシーのデフォルト URL には、サーバ名は含まれません。 前記のパラメータの値が yes に設定されている場合、Web エージェントは、パスワード ポリシーに存在する URL にユーザをリダイレクトします。

次の手順に従ってください:

1. ConstructFullPwsvcURl パラメータの設定のガイドとして、以下の表の例を使用してください。

変更後:	**管理 UI 内のパスワード ポリシーにこの URL を追加します。**	**ConstructFullPwsvcURl パラメータの値を次に設定します。**

変更後:	管理 UI 内のパスワード ポリシーにこの URL を追加します。	ConstructFullPwsvcURl パラメータの値を次に設定します。
特定のサーバ上でパスワード サービスをホストします。	http://server_name.example.com:80/siteminderagent/forms/smpwservices.fcc	いいえ
相対 URL を使用して、Web エージェントと同じサーバ上のパスワード サービスをホストします。	siteminderagent/forms/smpwservices.fcc	いいえ
FQDN を使用して、Web エージェントと同じサーバ上のパスワード サービスをホストします。	siteminderagent/forms/smpwservices.fcc	はい

FCC パスワード サービスを伴う SecureID 認証の設定

認証方式として SecureID を使用し、以下の両方の条件が環境内に存在する場合は、管理 UI を使用して SecureID HTML フォーム テンプレートを変更する必要があります。

- FCC パスワード サービス機能が設定されます。
- Web エージェントの SecureUrls パラメータの値が yes に設定されます。

SecureID はパスワード サービスを使用して実装されます。認証方式のテンプレートを変更する必要があるのはこのためです。

FCC パスワード サービスを使用して SecureID 認証を設定するには、以下の例に示されるように、SecureID テンプレートの［ターゲット］フィールドに smpwservices.fcc ファイルのパスを追加します。

```
/siteminderagent/forms/smpwservices.fcc
```

FCC でのユーザによるパスワード変更を有効にする方法

ユーザが必要に応じていつでも自分のパスワードを変更できるように SiteMinder の FCC パスワード サービス機能を設定できます。

注: SiteMinder Web エージェント設定で SecureURLs パラメータの値も No に設定されている場合のみ、以下のプロセスを実行します。

FCC でのユーザによるパスワード変更を有効にするには、以下の手順に従います。

1. ユーザ ディレクトリにパスワード ポリシーをサポートする属性が含まれていることを確認します。
2. 管理 UI を使用して以下のタスクを実行します。
   a. FCC ベースのパスワード ポリシーを作成し、対象のリソースを保護します。
   b. 権限のあるユーザがパスワードを変更できるようにパスワード ポリシーを設定します。
3. 以下が含まれるパスワード変更 URL を作成します。
   - ログオン サーバの FQDN （例： http:logonserver.example.com）
   - FCC ベースのパスワード サービスの URI （例： siteminderagent/forms/smpwservices.fcc?）
   - SiteMinder エージェントの名前（SMAGENTNAME）
   - 以下の*いずれか*のターゲット URL
   - FCC ページに埋め込まれたパスワード変更 URL については、次の例に示すように（SMAGENTNAME）および（TARGET）セクションに対して相対値を使用します。

     ```
     <a
     href="http:logonserver.example.com/siteminderagent/forms/smpwservices.fcc
     ?SMAUTHREASON=

     34&SMAGENTNAME=$$smencode(smagentname)$$&TARGET=$$smencode(target)$$">Cha
     nge Password</font></a>
     ```

   - FCC のページに埋め込まれ*ていない*パスワード変更 URL については、（SMAGENTNAME）セクションの SiteMinder エージェントの名前をハード コーディングします。 その後に、次の例に示すように（TARGET）セクションの完全修飾ドメイン名の値をハード コーディングします。

```
<a
href="http://logonserver.example.com/siteminderagent/forms/smpwservices.f
cc?SMAUTHREASON=34&SMAGENTNAME=Agent1&TARGET=https://logonserver.example.
com/protected/myprotectedpage.html">Change Password</font></a>
```

4. 1 つ以上の保護されていない Web ページ内のリンクとして、パスワード変更 URL （手順 3）を埋め込みます。
5. 以下の手順でパスワード変更をテストします。
   a. 手順 3 で作成したパスワード変更リンクがある Web ページを表示します。
   b. パスワード変更リンクをクリックします。

      パスワード変更フォームが表示されます。
   c. パスワード変更フォームに入力し、それをサブミットします。

      パスワードの変更に成功した場合、保護されているターゲット リソースへのリンクがある確認ページが表示されます。
   d. リンクをクリックし、リソースが表示されることを確認します。
   e. ブラウザを閉じて再度開きます。 新しいパスワードを使用して、保護されているリソースへのアクセスを試みます。

      新しいパスワードでリソースをアクセスできれば、パスワード変更は成功です。

FCC でのユーザによるパスワード変更を有効にする方法（SecureURLs=Yes）

ユーザが必要に応じていつでも自分のパスワードを変更できるように SiteMinder の FCC パスワード サービス機能を設定できます。

注: SiteMinder Web エージェント設定で SecureURLs パラメータの値も yes に設定されている場合のみ、以下のプロセスを実行します。

FCC でのユーザによるパスワード変更を有効にするには、以下の手順に従います。

1. ユーザ ディレクトリにパスワード ポリシーをサポートする属性が含まれていることを確認します。
2. 管理 UI を使用して以下のタスクを実行します。
   a. FCC ベースのパスワード ポリシーを作成し、対象のリソースを保護します。
   b. 権限のあるユーザがパスワードを変更できるようにパスワード ポリシーを設定します。
   c. ValidTargetDomain パラメータの値を、保護するターゲット リソースのドメインに設定します。
3. 以下が含まれるパスワード変更 URL を作成します。
   - ログオン サーバの FQDN （例： http:logonserver.example.com）
   - FCC ベースのパスワード サービスの URI （例： siteminderagent/forms/smpwservices.fcc?）
   - SiteMinder エージェントの名前（SMAGENTNAME）
   - 以下の*いずれか*のターゲット URL
   – FCC ページに埋め込まれたパスワード変更 URL については、次の例に示すように（SMAGENTNAME）および（TARGET）セクションに対して相対値を使用します。

```
<a
href="http:logonserver.example.com/siteminderagent/forms/smpwservices.fcc
?SMAUTHREASON=

34&SMAGENTNAME=$$smencode(smagentname)$$&TARGET=$$smencode(target)$$">Cha
nge Password</font></a>
```

- FCC のページに埋め込まれて*いない*パスワード変更 URL については、（SMAGENTNAME）セクションの SiteMinder エージェントの名前をハード コーディングします。 その後に、次の例に示すように（TARGET）セクションの完全修飾ドメイン名の値をハード コーディングします。

  ```
  <a
  href="http://logonserver.example.com/siteminderagent/forms/smpwservices.f
  cc?SMAUTHREASON=34&SMAGENTNAME=Agent1&TARGET=https://logonserver.example.
  com/protected/myprotectedpage.html">Change Password</font></a>
  ```

4. 1 つ以上の保護されていない Web ページ内のリンクとして、パスワード変更 URL （手順 3）を埋め込みます。
5. Web サーバで以下のファイルを開きます。

   web_agent_home/samples/forms/smpwservices.fcc

   a. 次の行を検索します。

      ```
      @smpwselfchange=0
      ```

   b. 以下の例のように、この行の最後の値を 0 から 1 に変更します。

      ```
      @smpwselfchange=1
      ```

   c. smpwservices.fcc ファイルを保存して閉じます。
6. 手順 3 で作成した URL を、保護されていない Web ページにリンクとして埋め込みます。
7. 以下の手順でパスワード変更をテストします。

   a. 手順 3 で作成したパスワード変更リンクがある Web ページを表示します。

   b. パスワード変更リンクをクリックします。

      パスワード変更フォームが表示されます。

   c. パスワード変更フォームに入力し、それをサブミットします。

      パスワードの変更に成功した場合、保護されているターゲット リソースへのリンクがある確認ページが表示されます。

   d. リンクをクリックし、リソースが表示されることを確認します。

   e. ブラウザを閉じて再度開きます。 新しいパスワードを使用して、保護されているリソースへのアクセスを試みます。

      新しいパスワードでリソースをアクセスできれば、パスワード変更は成功です。

SiteMinder X.509 証明書および基本認証方式を使用する場合にユーザによるパスワード変更を有効にする方法

ユーザ本人によるパスワード変更を許可するように SiteMinder の FCC パスワード サービス機能を設定できます。 SiteMinder X.509 証明書および基本認証方式では、HTTPS プロトコルで始まるパスワード変更 URL が必要です。

次の手順に従ってください:

1. ユーザ ディレクトリにパスワード ポリシーをサポートする属性が含まれていることを確認します。
2. 管理 UI を使用して以下のタスクを実行します。
   a. FCC ベースのパスワード ポリシーを作成し、対象のリソースを保護します。
   b. 権限のあるユーザがパスワードを変更できるようにパスワード ポリシーを設定します。
3. 以下が含まれるパスワード変更 URL を作成します。
   - HTTPS スキーム（プロトコル）
   - ログオン サーバの FQDN （例： http:logonserver.example.com）
   - FCC ベースのパスワード サービスの URI （例： siteminderagent/forms/smpwservices.fcc?）
   - SiteMinder エージェントの名前（SMAGENTNAME）
   - 以下の*いずれか*のターゲット URL
   - FCC ページに埋め込まれたパスワード変更 URL については、次の例に示すように（SMAGENTNAME）および（TARGET）セクションに対して相対値を使用します。

     ```
     <a
     href="https:logonserver.example.com/siteminderagent/forms/smpwservices.fcc?SMAUTHREASON=
     34&SMAGENTNAME=$$smencode(smagentname)$$&TARGET=$$smencode(target)$$">Change Password</font></a>
     ```

   - FCC のページに埋め込まれて*いない*パスワード変更 URL については、（SMAGENTNAME）セクションの SiteMinder エージェントの名前をハード コーディングします。 その後に、次の例に示すように（TARGET）セクションの完全修飾ドメイン名の値をハード コーディングします。

```
<a
href="https://logonserver.example.com/siteminderagent/forms/smpwservices.
fcc?SMAUTHREASON=34&SMAGENTNAME=Agent1&TARGET=https://logonserver.example
.com/protected/myprotectedpage.html">Change Password</font></a>
```

4. 1 つ以上の保護されていない Web ページ内のリンクとして、パスワード変更 URL （手順 3）を埋め込みます。
5. 以下の手順でパスワード変更をテストします。
   a. 手順 3 で作成したパスワード変更リンクがある Web ページを表示します。
   b. パスワード変更リンクをクリックします。

      パスワード変更フォームが表示されます。
   c. パスワード変更フォームに入力し、それをサブミットします。

      保護されているターゲット リソースへのリンクがある確認ページが表示されます。
   d. リンクをクリックし、リソースが表示されることを確認します。
   e. ブラウザを閉じて再度開きます。 新しいパスワードを使用して、保護されているリソースへのアクセスを試みます。

      新しいパスワードでリソースをアクセスできれば、パスワード変更は成功です。

第 14 章: シングル サインオン（SSO）

このセクションには、以下のトピックが含まれています。

OPTIONS メソッドを使用するリソースへの自動アクセス許可

SiteMinder Web エージェントでは、OPTIONS メソッドを使用するリソースに対してアクセスが試行されると、認証済みユーザでもクレデンシャルが要求されます。OPTIONS メソッドを使用するリソースの例には以下があります（これに限られるわけではありません）。

- Microsoft® Word 文書
- Microsoft® Excel® スプレッドシート

この要求は、リソースと関連付けられたアプリケーションが OPTIONS メソッドを使用してリクエストを Web サーバに送信するために発生します。このリクエストには SiteMinder cookie が含まれていないため、Web エージェントでは要求が発行されます。

これらのリソースに対する認証要求を防ぐ方法

1. 以下のパラメータの値を yes に設定します。

   autoauthorizeoptions

   HTTP OPTIONS メソッドを使用するリソースに対する全てのリクエストを自動的に許可します。

   このパラメータの値を yes に設定した場合、PersistentCookies パラメータの値は no に設定します。

   制限： yes、no

2. PersistentCookies パラメータの値を no に設定します。

単一ドメインでのシングル サインオンの仕組み

SiteMinder には、単一および複数の cookie ドメインで使用するシングル サインオン機能が用意されています。 ユーザは、シングル サインオン環境内であれば移動の際に再認証する必要がないため、この機能は、異なる Web サーバおよびプラットフォーム間でのアプリケーションの使用を簡略化し、また、ユーザ エクスペリエンスを向上させます。

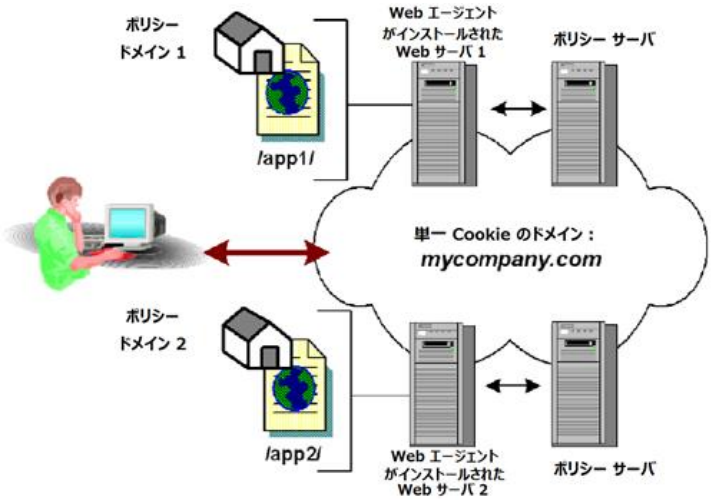
単一ドメイン環境では、すべてのリソースが 1 つの cookie ドメインに存在します。各 Web エージェントの設定で同じ cookie ドメインを指定すれば、同じ cookie ドメイン内の複数の Web エージェントに対してシングル サインオンを設定できます。

シングル サインオンが有効な場合、以下の手順が使用されます。

1. ユーザが一度認証します。
2. Web エージェントは成功した認証をキャッシュし、ユーザのブラウザ宛てにシングル サインオン cookie を発行します。
3. シングル サインオン cookie はセッション情報を提供します。その結果、ユーザは再認証なしで以下のタイプのリソースにアクセスできます。
   - 他のレルムにある保護されたリソース（ただし、保護レベルが*同等*または*それ以下*であるもの）
   - この cookie ドメイン内の別の Web サーバ

   保護レベルが*さらに高い*リソースにアクセスしようとするユーザは、アクセスが付与される前に再認証を受ける必要があります。

以下の図は、単一 cookie ドメインにおけるシングル サインオンを示しています。

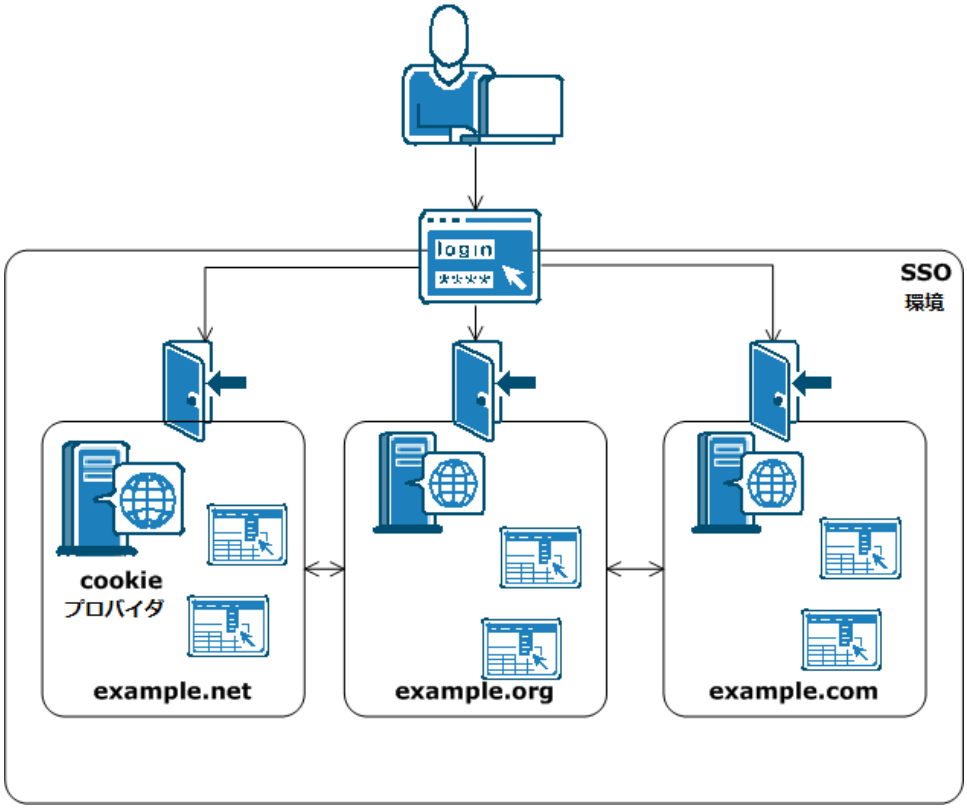


注: 複製ユーザ ディレクトリと共に非複製ポリシー ストアを使用している場合、ユーザ ディレクトリ名は、すべてのポリシー ストアで同一である必要があります。また、セッション チケットを暗号化するセッション チケット キーは、SSO 環境におけるすべてのキー ストアに対して同一でなければなりません。 セッション チケットにより、有効なユーザ セッションの持続時間が決定されます。

複数のドメインにおけるシングル サインオン

シングル サインオンを使用しないと、ユーザは、異なる cookie ドメインの別個のサーバ上にある別のアプリケーションやリソースにアクセスするときに、何度もログオンして認証情報を入力することが必要になります。複数の cookie ドメイン間でシングル サインオン情報を渡す機能を使用すると、ある cookie ドメインのサイトで認証されたユーザは、再認証を要求されることなく、別の cookie ドメインのサイトに移動できます。 このようなシームレスな移動により、ユーザが関連サイトを使用するときの利便性が向上します。

以下の図に、複数の cookie ドメインにおけるシングル サインオンを示します。



複数の Cookie ドメインにおけるハードウェア ロード バランサおよびシングル サインオン

SiteMinder では、cookie プロバイダとして設定された SiteMinder Web エージェントを使用して、複数の cookie ドメインにおけるシングル サインオンを実装します。

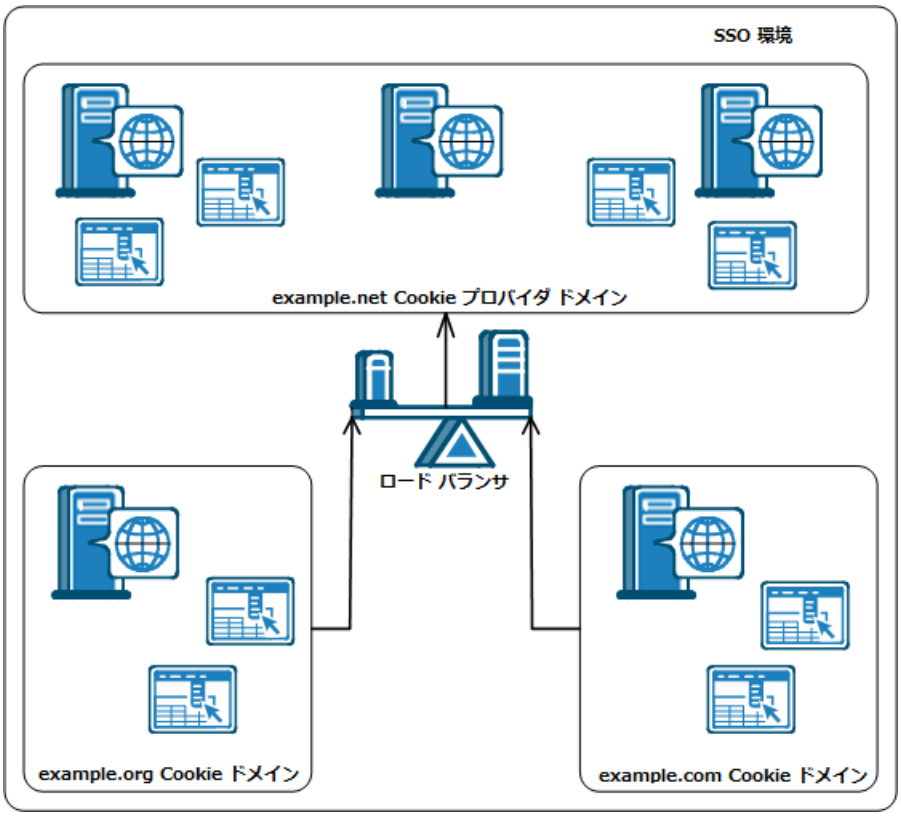
cookie プロバイダの Web エージェントが存在する cookie ドメインのことを、cookie プロバイダ ドメインといいます。 シングル サインオン環境において、他の cookie ドメインにあるその他すべての Web エージェントは、1 つの cookie プロバイダを参照しています。

SiteMinder Cookie プロバイダは以下の手順を使用して動作します。

1. ユーザがシングル サインオン環境内のドメインの保護されているリソースを要求すると、認証情報が要求されます。
2. ユーザが認証された場合、以下の Cookie がユーザのブラウザ内で設定されます。
   - ユーザが認証されたドメインのローカル Cookie
   - Cookie プロバイダが Cookie を設定します。
3. ユーザは以下のいずれかのイベントが発生するまで、再認証を要求されることなく、シングル サインオン環境内のドメイン間を移動できます。
   - ユーザのセッションがタイムアウトになる。
   - ユーザがセッションを終了する（通常はブラウザを閉じる）。

シングル サインオン環境内のエージェントは負荷分散を使用するか。

SSO 環境内のエージェントはすべて一つの Cookie プロバイダ ドメインを参照する必要があります。 Cookie プロバイダ ドメインの Web サーバと SSO 環境内の他の Cookie ドメインの間にロード バランサを追加します。以下の図に例を示します。



example.org Cookie ドメイン内の Web エージェント、および example.com Cookie ドメイン内の Web エージェントは両方とも、example.net の同じ Cookie プロバイダ ドメインを参照しています。 ロード バランサは、example.net Cookie プロバイダ ドメイン内のすべての Web サーバ間でトラフィックを均等に分散させます。

注: 複数の Cookie ドメインにわたって SSO を実装するために同じユーザ ディレクトリを使用する必要はありません。 ただし、複製ユーザ ディレクトリと共に非複製ポリシー ストアを使用している場合、ユーザ ディレクトリ名は、すべてのポリシー ストアで*同一*である必要があります。 また、セッション チケットを暗号化するセッション チケット キーは、SSO 環境におけるすべてのキー ストアに対して同一でなければなりません。 セッション チケットにより、有効なユーザ セッションの持続時間が決定されます。

シングル サインオンと認証方式の保護レベル

シングル サインオンを使用することで、あるレルムの認証済みユーザは、2 つ目のレルムが同等かそれ以下の保護レベルの認証方式で保護されていれば、再認証せずにその 2 つ目のレルムのリソースにアクセスすることができます。 ユーザが、より高い保護レベルの認証方式で保護されているリソースにアクセスしようとすると、SiteMinder は、認証情報の再入力を求めるプロンプトをユーザに表示します。

SiteMinder では、管理者が 管理 UI を使用して認証方式に保護レベルを割り当てることができます。 保護レベルの範囲は 1 から 20 です。1 が最も安全性が低く、20 が最も安全性が高くなります。 これらの保護レベルにより、管理者はシングル サインオン環境下のセキュリティおよび柔軟性に関する基準を追加して、認証方式を実装することができます。

たとえば、すべてのユーザに提供されている一連のリソースでは、保護レベル 1 の基本認証方式が使用されているとします。また、会社の重役のみに提供されている別の一連のリソースでは、保護レベル 15 の X.509 証明書方式が使用されています。 その場合に、ユーザがベーシック方式で認証を受けた後、証明書方式で保護されたリソースにアクセスを試みると、そのユーザは再認証を求められます。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

シングル サインオンとエージェント キー管理

Web エージェントは、Web エージェント間で情報を渡す cookie の暗号化と復号化にキーを使用します。 エージェントは、SiteMinder cookie を受け取ると、キーを使用して cookie の内容を復号化します。 キーは、ポリシー サーバと通信するすべての Web エージェントで、同じ値に設定してください。

キーの安全性を確保するため、ポリシー サーバは、これらのキーを生成し、暗号化して、SiteMinder 環境下のすべての Web エージェントに配布することができます。 同じキー ストアにすべてのキー情報を保持して共有する大規模な SiteMinder インストール環境でも、自動キー変換によって、エージェント キー管理を簡単に実施することができます。 自動キー変換により、キーの完全性も確保されます。

シングル サインオンの設定方法

シングル サインオン環境を設定するには、次の手順に従ってください：

1. シングル サインオン環境の Cookie ドメインを指定します。
2. シングル サインオン環境において、Cookie プロバイダ ドメインとして動作する Cookie ドメインを選択します（手順 1）。
3. エージェントが中央設定を使用する場合は、［エージェント設定オブジェクト］を開きます（管理 UI を使用）。ローカル設定を使用するエージェントについては、Web エージェント設定ファイル（各 Web サーバ上にあります）を開きます。
4. Cookie プロバイダであるエージェントについては、以下の手順で示すように設定パラメータを変更します。
   a. セキュリティ強化のために Cookie プロバイダ機能を制限します (P. 280)。
   b. Cookie プロバイダ リプレイ攻撃を防御します (P. 281)。
   c. RequireCookies パラメータの値が yes であることを確認します (P. 282)。
   d. （オプション）設定されたセッション タイムアウトまで Cookie を有効にしておくには、永続的 Cookie を有効にします (P. 283)。永続的 Cookie がない場合、ブラウザ Cookie は一時的です。 一時的 Cookie は *1* つのブラウザ セッションにのみ有効です。
   e. CookieDomain パラメータの値が、エージェントがインストールされたシステムのローカル Cookie ドメインを指定することを確認します (P. 284)。
   f. IP アドレスを検証するには、以下の*いずれか*のパラメータを設定します。
      - 永続的 Cookie を使用している場合は、PersistentIPCheck パラメータを設定します。
      - 一時的な Cookie を使用している場合は、TransientIPCheck パラメータを設定します。
5. （オプション）必要に応じて以下のシングル サインオン パラメータ設定を変更します。
   - セッション更新期間の変更
   - 複数ドメインにわたる安全な Cookie の設定

- 保護されていないリソースにおける Cookie プロバイダの無視
- POST 要求における Cookie プロバイダの無視
- シングル サインオンと併用する場合の SecureUrls の設定

6. 設定パラメータを設定した SSO 環境内の他のすべてのエージェント（Cookie プロバイダでないすべてのエージェント）の場合、以下の手順に従います。

   a. CookieProvider パラメータの値を Cookie プロバイダ ドメインの名前に設定します。 Cookie プロバイダとして動作しているエージェントをホストする Web サーバの完全修飾ドメイン名を使用します (P. 292)。

      以下の例で示す構文を使用します。

      http://*server.example.com:port*/siteminderagent/SmMakeCookie.ccc

      注: 前の例に示すように、Cookie プロバイダ名には .ccc 拡張子が必要です。

   b. セキュリティ強化のために Cookie プロバイダ機能を無効にします (P. 293)。

7. エージェント設定ファイルを修正することによってパラメータを編集した場合は、Web サーバを再起動して変更を反映させます。

Cookie プロバイダ機能の制限

すべてのエージェントにはデフォルトで有効になっている Cookie プロバイダ機能があります。盗んだ SiteMinder SSO Cookie を持つ不正なユーザは、Cookie プロバイダを利用してあるドメインのセッション Cookie を使用し、別の Cookie ドメインのセッション Cookie を偽造しようとする場合があります。偽造されたこれらのセッション Cookie により、保護されている SSO ドメインへの不正なアクセスが許可される可能性があります。

以下のパラメータを使用して、盗まれた SSO Cookie による Cookie プロバイダの利用と、セッション Cookie 偽造の可能性を排除できます。

LimitCookieProvider

Cookie プロバイダとして動作する SiteMinder エージェントが Cookie プロバイダ SET 要求（.ccc リソース）を処理する方法を指定します。 このパラメータの値が Yes の場合、Cookie が Cookie プロバイダのドメインに存在しない限り、SET 要求は無視されます。Cookie プロバイダは新しい Cookie を設定せずに、ユーザを TARGET URL にリダイレクトします。このパラメータの値が No の場合、SET 要求は処理され、TARGET URL へのリダイレクト中に新しい Cookie が常に設定されます。

デフォルト： No

デフォルト:（ユーザが smpolicy-secure.xml を使用してポリシー ストアを作成した後） Yes。

Cookie プロバイダとして動作するエージェントと、SSO 環境で動作する他のエージェントは、最適なセキュリティのために特定の設定を必要とする場合があります。

たとえば、ユーザの SSO 環境に 3 つのドメインが含まれると仮定します。example.com 内の Cookie プロバイダ、および 2 つの SSO ドメイン（example.org と example.net） 次の表で、各ドメインのエージェント設定について説明します。

Example.com（Cookie プロバイダ ドメイン）	Example.org（SSO Cookie ドメイン）	Example.net（SSO Cookie ドメイン）
CCCExt = .ccc	**CookieProvider** = http://server1.example.com:80/siteminderagent/SmMakeCookie.ccc	**CookieProvider** = http://server1.example.com:80/siteminderagent/SmMakeCookie.ccc

IgnoreExt = (verify that list of extensions includes .ccc)	**CCCExt** = .ccc	**CCCExt** = .ccc
EnableCookieProvider = yes	**IgnoreExt** = (verify that list of extensions includes .ccc)	**IgnoreExt** = (verify that list of extensions includes .ccc)
LimitCookieProvider = yes	**EnableCookieProvider** = no	**EnableCookieProvider** = no
TracksSessionDomain = yes	**TracksSessionDomain** = yes	**TracksSessionDomain** = yes
TrackCPSessionDomain = yes		

Cookie プロバイダ リプレイ攻撃の防御

次のパラメータを使用して、リプレイ攻撃への脆弱性から Cookie プロバイダを守ることができます。

TrackCPSessionDomain

セッション Cookie の Cookie ドメインが Cookie プロバイダの Cookie ドメインに一致することを検証します。 Cookie ドメインが異なる場合はリプレイ攻撃を示している可能性があります。

デフォルト： No（Cookie プロバイダのドメインは検証されません）

Cookie プロバイダ リプレイ攻撃を防ぐには、TrackCPSessionDomain パラメータの値を yes に設定します。

エージェントは Cookie ドメインを比較し、ドメインが一致しないときには要求を拒否します。

シングル サインオン用 RequireCookies パラメータの設定

以下のパラメータを使用して、SiteMinder が Cookie を必要とするかどうかを制御できます。

RequireCookies

SiteMinder が Cookie を必要とするかどうかを指定します。SiteMinder では、以下の機能のために Cookie が必要です。

- シングル サインオン環境の保護。
- セッション タイムアウトの適用。
- アイドル タイムアウトの適用。

このパラメータの値が Yes の場合、エージェントは HTTP 要求を処理するために以下のいずれかの Cookie を必要とします。

- SMCHALLENGE
- SMSESSION

このパラメータの値が No の場合、次の条件が発生する可能性があります。

- ユーザに対して、予期せず認証情報の認証が行われます。
- タイムアウトが厳密に適用されません。

重要: エージェントが Cookie を必要とする場合は、ブラウザ内で HTTP Cookie を受け入れるようにユーザに指示してください。そうしない場合、ユーザはすべての保護リソースへのアクセスを拒否されます。

デフォルト： Yes

Cookie を必要とするには、RequireCookies パラメータの値を yes に設定します。

シングル サインオン用永続的 Cookie の有効化

複数のブラウザセッションでシングル サインオンを使用するには、永続的な Cookie を使用します。 以下の手順では、永続的な Cookie の考えられる 1 つの使用法について説明します。

1. ユーザは SiteMinder で認証しますが、SiteMinder セッションが期限切れになる前に、ブラウザ セッションを終了します。
2. ユーザは後で新規ブラウザ セッションを開始しますが、永続的な Cookie はシングル サインオン機能を維持します。

永続的な Cookie は、設定された最大セッション タイムアウトに 7 日間*加えた*期間で有効です。Cookie が期限切れになった後、多くのブラウザは、Web ブラウザの Cookie ファイルを削除します。 一部のブラウザは永続的な Cookie を異なる方法で処理する場合があります。

次の手順に従ってください:

1. PersistentCookies パラメータに yes を設定します。

   SMSESSION Cookie は永続的です。

2. TransientIDCookies パラメータを no に設定します。

   SMIDENTITY Cookie は永続的です。

Cookie ドメインの指定

CookieDomain パラメータは、ユーザがエージェントをインストールしたサーバの Cookie ドメインを定義します。 以下のパラメータの設定によりドメインを変更できます。

CookieDomain

エージェントの Cookie ドメインを定義します。 少なくとも 2 つのピリオドが含まれている、完全修飾ドメイン名を使用します。 たとえば、.example.com という Cookie ドメインの設定は以下のサーバと一致します。

- w1.example.com
- w2.example.com
- w3.sales.example.com

このドメイン内のすべての Web サーバは、ブラウザと Cookie を送受信できます。 同一 Cookie ドメイン内のサーバは、Cookie を使用してユーザの認証情報を確認します。

パラメータ値が none の場合、エージェントは自身のサーバ用のみの Cookie を生成します。 たとえば、myserver.example.com です。

値が空白の（またはローカル設定ファイルに""が含まれている）場合、エージェントは HTTP_HOST ヘッダのドメイン情報を使用します。 その後、エージェントはこの設定を使用して値を CookieDomainScope パラメータに置きます。

デフォルト： 空白

例： .example.com

制限： この値は、大文字と小文字が区別されます。 前の例で示したように、この値には少なくとも 2 つのピリオドが含まれる完全修飾ドメイン名が必要です。

注: この値では、大文字と小文字が区別されます。

次の手順に従ってください:

1. CookieDomain パラメータの値を設定します。
2. （オプション）CookieDomainScope パラメータの値を設定します。

   CookieDomainScope

   ドメイン名のセクション（ピリオドで区切られた文字）の数を指定します。

   この値が 0（デフォルト）に設定されている場合、エージェントはサーバ専用の Cookie を作成することなく、そのホストに最も特定された Cookie ドメインを選択します。 これは、Cookie ドメイン myserver.example.com に対応するドメインが example.com であり、myserver.metals.example.org に対応する ドメイン が .metals.example.org であることを意味します。

   CookieDomainScope パラメータが 2 に設定されている場合、Cookie ドメインはそれぞれ .example.com と .example.org になります。

   デフォルト： 0

   例: ユーザの Cookie ドメインが division.example.com であると仮定します。 server.division.example.com の Cookie ドメインのスコープを設定するには、CookieDomainScope パラメータの値を 3 に設定します。

シングル サインオン環境用の IP アドレス検証の有効化

無許可のシステムでは、パケットを監視して Cookie を不正に入手し、その Cookie を使って別のシステムにアクセスすることができます。 無許可のシステムによるセキュリティ侵害を防止するために、永続的な Cookie と一時的な Cookie を使用して、IP チェックを有効または無効にすることができます。

IP チェック機能では、エージェントが現在の要求に含まれている IP アドレスに対して最後の要求の Cookie に格納された IP アドレスを比較する必要があります。 IP アドレスが一致しない場合、エージェントは要求を拒否します。

IP チェックを実装する目的で使用される 2 つのパラメータは、PersistentIPCheck と TransientIPCheck です。これらを次のように設定します。

- PersistentCookies を有効にした場合、PersistentIPCheck を yes に設定します。
- PersistentCookies を有効にしなかった場合、TransientIPCheck を yes に設定します。

SiteMinder ID Cookie は、IP チェックの影響を受けません。

セッション更新期間の変更

以下のパラメータを使用して、Web エージェントが cookie プロバイダにリクエストをリダイレクトして新しい cookie を設定する間隔を指定することができます。

SessionUpdatePeriod

新しい cookie を設定する目的で、Web エージェントが要求を cookie プロバイダにリダイレクトする頻度(秒単位)を指定します。マスタ cookie を更新すると、SiteMinder セッションのアイドル タイムアウトが原因でその cookie が期限切れになる確率を低下させることができます。

デフォルト： 60

セッション更新期間を変更するには、以下の手順に従います。

1. CookieProvider パラメータが定義されていることを確認します。
2. 目的の間隔を反映するように、SessionUpdatePeriod パラメータの秒数を変更します。

   セッション更新期間が変更されます。

複数ドメインにわたる安全な Cookie の設定

UseSecureCookies パラメータの設定により、それらの間の接続が安全な（HTTPS）場合に*ローカル* cookie を要求ブラウザ セッションに返すためにのみ Web エージェントが設定されます。Web エージェントが cookie プロバイダとしても設定されている場合、UseSecureCookies は他の cookie ドメインのリソースへのアクセスに対するリダイレクトされた要求には適用されません。

cookie プロバイダとして機能する Web エージェントが、安全な cookie を使用するようにも設定されている場合に、別の cookie ドメインの Web エージェントにのみ cookie を返すように設定するには、UseSecureCookies を有効にし、以下のパラメータも設定する必要があります。

UseSecureCPCookies

UseSecureCPCookies が Yes に設定されていると、cookie プロバイダは、セキュア cookie の使用も設定されている（つまり、UseSecureCookies も有効である）別の cookie ドメイン内の Web エージェントにのみ cookie を送信します。

この設定と UseSecureCookies が両方とも有効な場合、複数ドメインのシングル サインオン環境内のユーザは、SSL の Web サーバから別の cookie ドメインの非 SSL の Web サーバに移動するときに、再認証する必要があります。セキュア cookie は、従来の HTTP 接続を介して渡すことはできません。

デフォルト： No

複数のドメインの SSL 接続に cookie を送るには、cookie プロバイダで UseSecureCookies と UseSecureCPCookies を yes に設定します。

詳細情報:

安全な Cookie の設定 (P. 107)

保護されていないリソースにおける Cookie プロバイダの無視

エージェントはデフォルトで、すべての要求を Cookie プロバイダに転送します。 保護なしリソースがある場合は、以下のパラメータを使用してネットワーク トラフィックを削減することができます。

IgnoreCPForNotprotected

保護されていないリソースの要求に関して、Cookie プロバイダがクエリを行わないようにします。 このパラメータを「No」に設定すると、Web エージェントによってすべての要求が Cookie プロバイダに送られます。 従来の（非フレームワーク）エージェントについては、このパラメータの値が Web エージェント ログ ファイルに表示されるように、Cookie プロバイダを設定します。

デフォルト： No

保護なしリソースが要求された場合に、エージェントが Cookie プロバイダに問い合わせないようにするには、IgnoreCPForNotprotected パラメータの値を yes に設定します。

POST 要求における cookie プロバイダの無視

以下のパラメータがこれらの動作を有効にします。

- この設定を使用すると、従来のエージェントが cookie プロバイダとして動作できます。
- この設定は、POST リクエストが Cookie プロバイダ（すべての環境）に送信されるのを防止します。

  LegacyCookieProvider

  エージェントが Cookie プロバイダに POST 要求を送信するかどうかを制御します。 エージェントが（Cookie プロバイダとして動作する）従来のエージェントに POST 要求を送信すると、リダイレクトされた要求は GET になります。 この変換はエラーを引き起こします。no に設定すると、エージェントは Cookie プロバイダに POST 要求を送信します。 yes に設定すると、エージェントは Cookie プロバイダに POST 要求を送信 *しません*。

  中央エージェント設定を使用している場合は、このパラメータをエージェント設定オブジェクトに追加します。このパラメータは、ローカル設定ファイル内に存在します。

  デフォルト： なし（送信された POST 要求）

LegacyCookieProvider パラメータの値を yes に設定し、以下の動作を有効にします。

- cookie プロバイダとして従来のエージェントを使用します。
- POST リクエストが cookie プロバイダに送信されないようにします。

シングル サインオンと併用する場合の SecureUrls の設定

シングル サインオン ネットワーク内に SecureUrls 機能をサポートする Web エージェントと SecureUrls 機能をサポートしない別のエージェントが存在する場合、ユーザが保護されたシングル サインオン リソースを要求すると内部サーバ エラー メッセージが表示されることがあります。

SecureUrls がサポートされる Web エージェントのログには、以下のように、サーバ エラーの理由が表示されます。

エラー： 要求を処理できません。SecureUrls が無効です。

注: シングル サインオン環境内のすべての Web エージェントについて、SecureUrls パラメータに同じ値を設定する必要があります。SiteMinder は、SecureUrls パラメータの値が異なる Web エージェント間の相互運用性をサポートしていません。

Cookie プロバイダの指定

SSO 環境の他のエージェントに Cookie プロバイダを使用させるには、以下のパラメータを使用して Cookie プロバイダとして動作しているエージェントの場所を指定します。

CookieProvider

Cookie プロバイダとして動作しているエージェントがある Web サーバの URL を指定します。

Cookie プロバイダはシングル サインオン環境内のエージェントです。 Cookie プロバイダは、その Cookie プロバイダが存在するローカル ドメインのブラウザ Cookie を設定します。この Cookie が設定された後、ユーザは再認証しなくてもシングル サインオン環境全体を移動できます。

次の例に示すように、Cookie プロバイダ名には .ccc 拡張子が必要です。

- IIS、Oracle iPlanet、および Domino Web サーバでは、以下の URL 構文を使用します。

  http://*server.domain:port*/siteminderagent/SmMakeCookie.ccc

- Apache および Apache ベースの Web サーバでは、以下の URL 構文を使用します。

  http://*server.domain:port*/SmMakeCookie.ccc

このパラメータは以下のパラメータにも影響します。

- CCCExt
- SessionUpdatePeriod

デフォルト： デフォルトなし

例： （IIS、Oracle iPlanet、および Domino Web サーバ）
http://server1.example.com:80/siteminderagent/SmMakeCookie.ccc

例： （Apache および Apache ベースの Web サーバ）
http://server1.example.com:80/SmMakeCookie.ccc

制限： このパラメータには完全修飾ドメイン名が必要です。

次の手順に従ってください:

1. CookieProvider パラメータに、Cookie プロバイダとして動作している Web サーバの URL を設定します。
2. CCCExt パラメータの値が .ccc に設定されていることを確認します。

3. IgnoreExt パラメータの値に .ccc 拡張子を追加します。
4. （任意）セッション更新期間を変更します。
5. SSO 環境内の Cookie プロバイダでないすべての Web エージェントに対して、手順 1 ～ 4 を繰り返します。

   これで、Cookie プロバイダが指定されます。

Cookie プロバイダの無効化

デフォルトでは、すべての SiteMinder エージェントは Cookie プロバイダとして動作できます。 この設定により SSO 環境の設定がより容易になります。 セキュリティ強化のために、組み込まれた Cookie プロバイダ機能を以下のパラメータを使用して無効にできます。

EnableCookieProvider

エージェントが Cookie プロバイダ（.ccc）からの要求を処理する方法を指定します。 このパラメータ値が Yes の場合、エージェントは要求を処理します。このパラメータ値が No の場合、エージェントは Cookie プロバイダからの要求を無視します。 エージェントは要求されたリソースへのアクセスを拒否します。 セキュリティを強化するには、このパラメータ値を No に設定します。

デフォルト： Yes

デフォルト:（ユーザが smpolicy-secure.xml を使用してポリシー ストアを作成した後） No。

Cookie プロバイダからの要求をエージェントが処理するのを防ぐには、EnableCookieProvider パラメータの値を No に設定します。

環境で SSO に Cookie プロバイダを使用しない場合は、すべてのエージェントについて、次の表に示すエージェント設定値を使用します。

すべてのエージェントの設定パラメータを以下の値に設定します。
EnableCookieProvider = no

第 15 章: 包括的ログアウト

このセクションには、以下のトピックが含まれています。

完全ログオフの仕組み

完全ログオフを使用すると、Web 開発者は、ユーザ セッションからユーザを完全にログオフさせることができます。これにより、ユーザは Web ブラウザを終了せずにセッションを終了できるようになり、無許可のユーザは開いているセッションを不正に制御できなくなるため、リソースが保護されます。

完全ログオフでは以下の手順が使用されます。

1. ユーザがボタンをクリックするとログ オフします。
2. Web エージェントは、作成されたカスタマイズ ログオフ ページにユーザをリダイレクトします。
3. Web エージェントは、ユーザのブラウザからセッション cookie と認証 cookie を削除します。
4. Web エージェントは、ローカル cookie ドメインと cookie プロバイダ ドメインからもセッション cookie を削除します。このドメインは、シングル サインオン環境用に指定したものです。
5. Web エージェントはポリシー サーバを呼び出して、セッション情報を削除するように指示します。

   ユーザは完全にログオフされます。

詳細情報:

Domino エージェントの完全ログオフ サポートの設定 (P. 404)

完全ログオフの設定

完全ログオフ機能では、以下のパラメータで作成するカスタム ログアウト ページが使用されます。

LogOffUri

カスタム Web ページの URI を指定して完全なログアウト機能を有効にします。 ユーザが正常にログオフした後にこのカスタム Web ページが表示されます。 ブラウザ キャッシュ内に格納できないようにこのページを設定します。設定しなかった場合、ブラウザは、ユーザをログ オフせずに、そのキャッシュからログアウト ページを表示する場合があります。 この状況が発生すると、不正なユーザにセッションの支配権を握る機会を与える可能性があります。

注: CookiePath パラメータが設定されているときは、LogOffUri パラメータの値が同じ cookie パスを指している必要があります。 たとえば、CookiePath パラメータの値が example.com に設定されている場合、LogOffUri は example.com/logoff.html を指している必要があります。

デフォルト：（CA SiteMinder Agent for SharePoint r12.0.3.0 *以外*のすべてのエージェント）デフォルトなし

制限： 複数の URI 値を指定できます。 完全修飾 URL は使用 *しないでください*。*相対* URI を使用します。

例： （CA SiteMinder Agent for SharePoint r12.0.3.0 *以外*のすべてのエージェント） /Web pages/logoff.html

次の手順に従ってください:

1. ユーザのログオフ用のカスタム HTTP アプリケーションを作成します。たとえば、ユーザを指定した URL にリダイレクトするための終了ボタンまたはサインオフ ボタンを追加します。
2. ログアウト ページを Web ブラウザにキャッシュできないように設定します。 この設定により、ページはブラウザのキャッシュではなく Web サーバから常に提供されるので、セキュリティが向上します。 たとえば、HTML ページの場合は、ページに次のようなメタタグを追加します。

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">

<META HTTP-EQUIV="Expires" CONTENT="-1">
```

重要: 一部の Web ブラウザは、メタ タグをサポート *していません*。代わりに、キャッシュ コントロール HTTP ヘッダを使用してください。

3. 以下の手順で LogOffUri パラメータを設定します。
   a. 必要に応じて、ポンド記号（#）を削除します。
   b. ユーザをログオフするカスタム HTTP ファイルの URI を入力します。 完全修飾 URL は使用 *しないでください*。

      完全なログアウト機能が設定されます。

詳細情報:

エージェント Cookie の Cookie パスの指定 (P. 110)

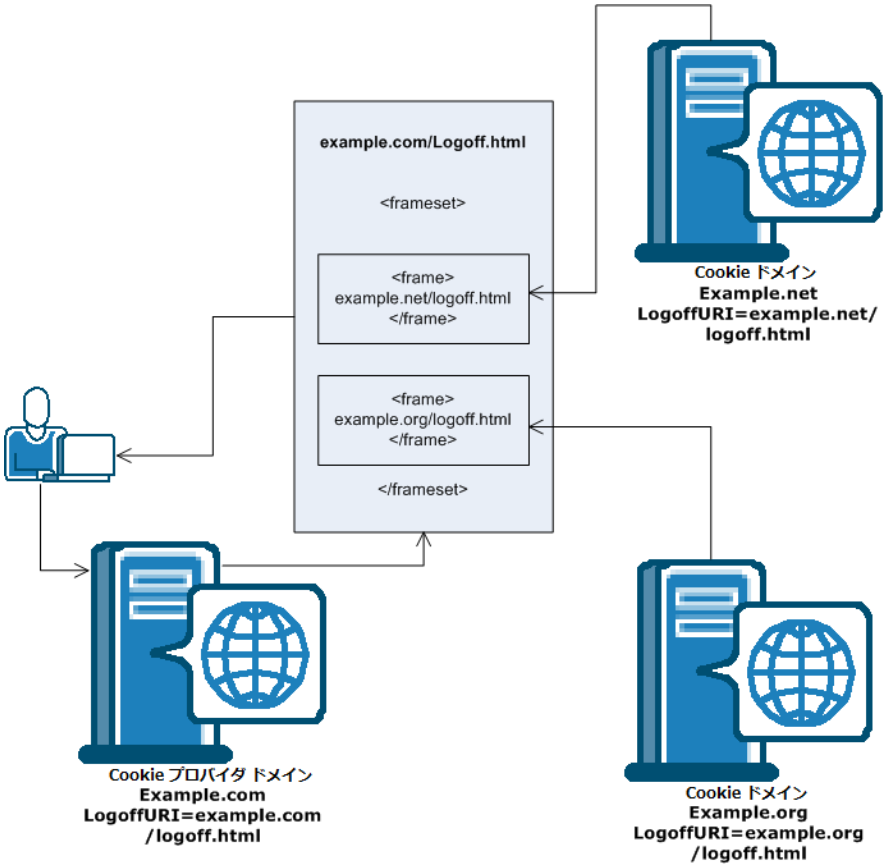
シングル サインオンでの完全ログオフの設定方法

シングル サインオン環境では、セッション cookie はローカル cookie ドメインと、Web エージェントに関連付けられた cookie プロバイダ ドメインからのみ削除されます。 複数の cookie ドメインにわたるシングル サインオンの場合、SiteMinder の完全ログオフ機能では、ユーザが訪問したすべての cookie ドメインにおいてユーザを自動的にログオフすることはしません。

複数の cookie ドメインにわたるログオフを設定するには、以下の手順を使用します。

1. SSO 環境内の他の cookie ドメイン用に別のフレーム（または iframes）を含む一元化されたログオフ ページを作成します。これらのフレームは、1x1 ピクセルのような小さいものにできます。
2. 手順 1 で作成されたログオフ ページの各フレームについて、関連する cookie ドメインのログオフ URI へのハイパーリンクを追加します。 たとえば、ほかに 2 つの cookie ドメイン（example.org と example.net）がある場合、以下の手順に従います。
   - 1 つのフレームに、example.org のログオフ URI へのハイパーリンクを追加します。
   - 別のフレームに、example.net のログオフ URI へのハイパーリンクを追加します。
3. cookie プロバイダ ドメインのログオフ URI が一元化されたログオフ ページを参照するように設定します。 Web サーバがこのログオフ ページをロードすると、一元化されたログオフ ページのフレームが他の cookie ドメインから各ログオフ ページを呼び出します。ユーザがすべての cookie ドメインから一度にログオフされます。

以下の図は、一元化されたログオフ ページを使用する例にを示しています。



注: ハイパーリンクは、<frame> タグではなく <iframe> タグ内に配置することもできます。

FCC フォームを使用した包括的ログアウトの設定

ユーザを認証するために FCC フォームを使用する場合、FCC フォームを使用して包括的ログアウトを設定できます。 このメソッドは LogoffUri パラメータの代替手段を提供します。

次の手順に従ってください:

1. テキスト エディタを使用して、ユーザを認証するために使用する .fcc ファイルを開きます。 FCC ファイルは、以下のディレクトリにあります。

   web_agent_home/samples/forms

   web_agent_home

   SiteMinder エージェントがインストールされているディレクトリを示します。

   デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）：C:¥Program Files¥CA¥webagent

   デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）：C:¥Program Files¥CA¥webagent¥win64

   デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）：C:¥Program Files (x86)¥webagent¥win32

   デフォルト（UNIX/Linux インストール）：/opt/ca/webagent

2. FCC ページの先頭（<_html> タグの前）に以下のテキストを追加します。

   ```
   @smlogout=true
   @target=http://server_name.example.com/directory/your_logout_page.html
   ```

   注: *your_logout_page* は、ユーザにログアウトしたことを通知するために作成したカスタム html ページを示します。

   FCC フォームを使用した包括的ログアウトが設定されました。

第 16 章: SSO セキュリティ ゾーン

このセクションには、以下のトピックが含まれています。

セキュリティ ゾーンの概要

ユーザは、同じ cookie ドメイン内に（単一のゾーンを表す）、または複数の cookie ドメインにまたがって（別々のゾーンを表す）、シングル サインオン セキュリティ ゾーンを定義できます。 その結果、ユーザには同じゾーンの中ではシングル サインオンが適用されますが、別のゾーンに入るときは、ゾーン間に定義された信頼関係に応じて認証情報を再要求されることもあります。トラステッド関係に含まれているゾーンでは、グループ内のいずれかのゾーンで有効なセッションを持つユーザには認証情報が再要求されません。

シングル サインオン セキュリティ ゾーンは、SiteMinder Web エージェントによって完全に実装されます。 各ゾーンは、別々の Web エージェント インスタンス上に存在する必要があります。 同じエージェント インスタンス上に複数のゾーンを作成することはできません。

セキュリティ ゾーンは、Web エージェントによって生成された cookie によって識別されます。デフォルトで、Web エージェントは、SMSESSION という名前のセッション cookie と SMIDENTITY という名前の ID cookie を生成します。 セキュリティ ゾーンの設定時に、Web エージェントは固有の名前を持つセッション cookie と ID cookie を生成して、ゾーンのアフィリエイト関係を cookie 名に反映させることができます。

セキュリティ ゾーンの定義

以下の用語は、シングル サインオン（SSO）セキュリティ ゾーンに関するものです。

CAC（Centralized Agent Configuration、中央エージェント設定）

Web エージェントが、ポリシー ストアに定義された Web エージェント設定オブジェクトから自身の設定プロパティを取得するためのメカニズムを示します。

Cookie プロバイダ

複数のドメインにまたがって Web エージェントにシングル サインオンを実装するためのメカニズムを示します。いずれかのドメインがマスタ ドメインとして指定されます。その他のドメインの Web エージェントは、マスタ ドメイン内の Web エージェントにリダイレクトされ、そのドメインの cookie が提供されます。

SSO（Single Sign-On、シングル サインオン）

一度認証されたユーザが、認証情報を再要求されないようにするためのメカニズムを示します。

SSO ゾーン

任意の識別子（ゾーン名）によって定義される SSO のサブセットを示します。単一の cookie ドメイン内でアプリケーション SSO をセグメント化するために使用されます。 同一 SSO ゾーン内のすべてのアプリケーションは、相互間で SSO を許可します。 SSO ゾーン間でのやり取りは、ゾーン信頼関係の定義に従って許可するかどうかが決まります。

トラステッド SSO ゾーン

SSO に関してローカル ゾーンから信頼されている外部ゾーンを示します。

セキュリティ ゾーンの利点

SSO セキュリティ ゾーン機能は、SiteMinder 管理者が同じ cookie ドメイン内にあるシングル サインオン環境をセグメント化する必要がある場合に使用することを目的としています。 たとえば、CA.COM というドメインがあるとします。標準の SiteMinder SSO 機能では、CA.COM 内の SiteMinder 保護下にあるすべてのアプリケーションは、SMSESSION cookie を使用してシングル サインオンを管理します。 以下のような、SSO セキュリティ ゾーンが存在しないシナリオを考えてみます。

1. ユーザがアプリケーション（APP1）にアクセスします。 ユーザは SiteMinder から認証情報を要求されます。SiteMinder にログインすると、SMSESSION cookie が作成されます。
2. ユーザは 2 つ目のアプリケーション（APP2）にアクセスし、SiteMinder から認証情報を再要求されます （ユーザは APP1 のユーザ認証情報を使用して APP2 にアクセスすることができないため、ルールに従って、SSO は適用されません）。ユーザがログインすると、新しい SMSESSION cookie が作成され、前の cookie は APP2 への新たなログイン セッションによって上書きされます。
3. ここでユーザが APP1 に戻ると、さらにもう一度認証情報を要求されます。これは、ユーザは元の APP1 セッションを失っており、かつ、APP1 は APP2 セッションを受け入れないためです。 したがって、APP1 と APP2 の間で SSO は適用されず、非常に面倒な状況になります。

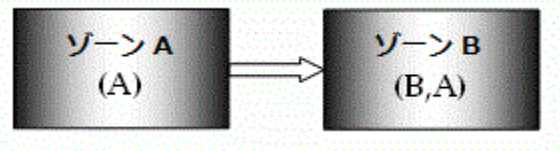
SSO セキュリティ ゾーンを使用すると、APP1 をゾーン Z1 に、APP2 をゾーン Z2 に置くことができます。 ここで、APP1 にログインすると Z1SESSION cookie が作成され、APP2 にアクセスすると Z2SESSION cookie が得られます。名前が異なるので、cookie は互いを上書きすることがなくなります。したがって、上の例のようにユーザはアプリケーション間を移動するたびにログインする必要がなくなり、アプリケーションごとに 1 回のログインで済むようになります。

SSO セキュリティ ゾーン機能が提供されるまでは、アプリケーションについて SSO を同様にグループ化する唯一の方法は、異なるネットワーク ドメインの作成を経て、異なる cookie ドメイン（CA1.COM、CA2.COM、その他）を作成し、各種のマルチ cookie ドメイン設定を cookie プロバイダと併用することでした。 これは、多くの企業にとって望ましくない方法です。複数のネットワーク ドメインを使用すると、相応の IT 保守およびサポートが必要になるからです。

セキュリティ ゾーンの基本ユースケース

シングル サインオンは、制御された基準に応じて、設定可能な信頼関係を持ついくつかのセキュリティ ゾーンに分けることができます。 たとえば、以下のゾーン A とゾーン B について考えてみます。

- ゾーン A の持つトラステッド ゾーンは、ゾーン A 自身のみです。
- ゾーン B は、ゾーン B 自身とゾーン A の 2 つのトラステッド ゾーンを持ちます。



上の図の信頼関係は、矢印で示されています。つまり、ゾーン A で確立されるユーザ セッションは、ゾーン B のシングル サインオンに使用できます。

この例では、ゾーン A は管理者専用ゾーンであるのに対して、ゾーン B は共通のアクセス ゾーンである可能性があります。 ゾーン A で認証された管理者は、再認証を要求されることなくゾーン B へのアクセス権を取得できます。 ただし、ゾーン B で認証されたユーザは、ゾーン A にアクセスしようとすると、再認証を要求されます。

別のゾーン内のユーザ セッションは、相互に独立しています。 ユーザがゾーン B で最初に認証された場合は、ゾーン B で再度認証されます。 2 つの異なるセッションが作成されます。 実際は、ユーザは両方のセッションで異なる ID を持ちます。 ユーザがゾーン A に戻ると、このゾーンで確立されたセッションが使用されます。

ユーザが、まだセッションを確立していないゾーンでシングル サインオンを使用して検証されると、どうなるかについて考えてみます。 ユーザがゾーン A で認証されてから、初めてゾーン B にアクセスすると、ゾーン A のセッション情報に基づいてユーザ セッションがゾーン B に作成され、場合によってはポリシー サーバによって更新されます。 ゾーン A のユーザ セッションは、ユーザがゾーン A に戻るまで更新されないことに注意してください。

セキュリティ ゾーン間のユーザ セッション

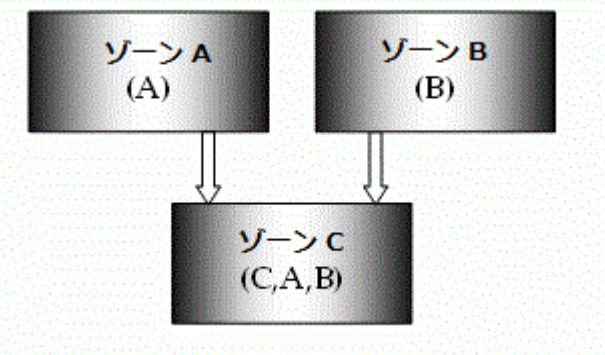
シングル サインオン セキュリティ ゾーンは、すべて同一ドメインに属する必要はありません。 実際は、ゾーンは複数のドメインにまたがることがあります。 ただし、Web エージェントが検索できるのは、ローカル cookie ドメイン内のトラステッド ゾーンの cookie のみです。 適切な cookie が見つからない場合は、Web エージェントは自身のゾーンのみで cookie プロバイダへのリダイレクトを続行します。

トラステッド ゾーンの順序

シングル サインオン セキュリティ ゾーンは、以下の 1 組のパラメータによって定義されます。

- セキュリティ ゾーン名
- トラステッド ゾーンの番号付きリスト

トラステッド ゾーンがリストされている順序は重要です。 以下の例について考えてみます。



この図では、ゾーン C はゾーン A とゾーン B の両方を信頼しています。ゾーン A とゾーン B はどちらも他のゾーンを信頼していませんが、すべてのゾーンが自身を信頼しています。

ユーザがゾーン C で要求を行うと、Web エージェントは、ゾーンがリストされている順に、トラステッド ゾーンでセッションまたは ID cookie を検索します。 この例では、ゾーン C には、C、A、および B が含まれたトラステッド ゾーン リストがあります。

発生する可能性があるイベントの順序を以下に示します。

1. Web エージェントは、ユーザがゾーン C でセッションを確立しているかどうかを最初に確認します。
2. セッションが見つからない場合は、Web エージェントは、ユーザがゾーン A でセッションを確立しているかどうかを確認します。
3. セッションが見つからない場合は、Web エージェントは、ユーザがゾーン B でセッションを確立しているかどうかを確認します。
4. 見つかった各 cookie からのセッションの指定は、正常にログインできるまで認証要求を処理するために使用されます。
5. 認証が正常に行われると、Web エージェントは許可プロセスに進みます。
6. cookie が見つからないか、cookie が認証を渡していない場合は、エージェントは通常どおりユーザに認証情報を要求します。

ゾーンにアクセスする順序によって、ユーザの操作が異なる場合があることに注意してください。 この例では、ユーザがゾーン B に最初にアクセスして、次にゾーン C にアクセスする場合は、ゾーン B でのユーザの ID がゾーン C でも使用されます。 ユーザがゾーン A に最初にアクセスして、次にゾーン B とゾーン C にアクセスする場合は、ユーザはゾーン C に移動する前にゾーン B で再認証を要求されるにもかかわらず、ゾーン A でのユーザの ID が使用されます。

これは、異なる最大セッション タイムアウトとアイドル セッション タイムアウトが指定されているセッションの有効期限が切れ始めたときにも当てはまります。 現在の例では、ゾーン A とゾーン C で有効な cookie を持つユーザは、最初にゾーン C の cookie へのアクセス権を取得します。ゾーン C の cookie の有効期限が切れると、ゾーン A の cookie が使用されます （有効期限が切れていない場合）。 そのため、ユーザの ID は、認証情報の要求が行われることなく、ゾーン C の ID からゾーン A の ID に変更されることがあります。

複数の Web エージェントが異なるトラステッド ゾーン リストを持つ場合でも、共通のトラステッド ゾーン名を使用します。 この場合は、エージェントは相互を暗黙的に信頼しますが、同一の外部ゾーンは信頼しません。 この機能を使用すると、シングル サインオンでアプリケーションをセグメント化できます。 Web エージェントは、シングル サインオン ゾーン名のみをサポートします。 このエージェントによって生成されるすべてのセッション、ID、および状態 cookie が、同一のシングル サインオン ゾーン名を使用します。 そのため、2 つのアプリケーションが同一のシングル サインオン信頼要件を共有していない場合、それらのアプリケーションは、それぞれ独自の Web エージェントとトラステッド ゾーン リストを持つ別々の Web サーバ上でホストする必要があります。

注: 外部ゾーンとは、特定の Web エージェントによってサポートされるゾーン以外のゾーンです。 たとえば、エージェントが SSOZoneName="Z1" として設定されている場合は、その他すべてのゾーンはこのエージェントにとって外部ゾーンになります。 これには、デフォルト ゾーンである「SM」も含まれます。

デフォルトのシングル サインオン ゾーンおよびトラステッド ゾーン リスト

セキュリティ ゾーン名を指定しない Web エージェント（SiteMinder 6.x QMR 5 より前のすべての Web エージェントなど）は、デフォルト ゾーンに属していると見なされます。 後方互換性を確保するために、デフォルト ゾーンは、ゾーン名 SM が付いていると暗黙的に想定されます。これによって、SiteMinder 12.52 Web エージェントは、設定変更せずにデフォルトで SMSESSION および SMIDENTITY をサポートすることができます。

トラステッド ゾーン リストを指定しない Web エージェントは、自身のシングル サインオン ゾーン（指定されたゾーン名、またはゾーン名が指定されていない場合はデフォルト ゾーン）のみを信頼します。

Web エージェントは、デフォルト ゾーンに加えて他のゾーンを信頼するように設定できます。 また、指定されたゾーン名を使用できますが、他のトラステッド ゾーンはリストできません。 追加のトラステッド ゾーンが指定されているかどうかに関係なく、エージェントは常に自身のゾーンを最初に信頼します。 デフォルト以外のゾーンを使用する Web エージェントがデフォルト ゾーンも信頼するようにするには、トラステッド ゾーン リストに「SM」を追加する必要があります。

複数のユーザ セッションによる要求処理

Web エージェントは、トラステッド ゾーンの順序でユーザ セッションを検索します。有効なユーザ セッションが見つかった場合は、Web エージェントはセッション情報を使用して、ユーザ要求を処理します。 セッションが見つからない場合は、Web エージェントは、信頼の順序で次に有効なユーザ セッションにフェールオーバします。

検査対象の別のセッションがある場合は、失敗した検証からのレスポンスは無視されます。 別のものがない場合は、通常どおり処理されます。 これは、Web エージェントが処理対象のトラステッド セッションを 3 つ見つけた場合に、最初の 2 つの検証に失敗すると、3 番目（最後）のセッションの検証からのレスポンスのみが処理されることを意味します。

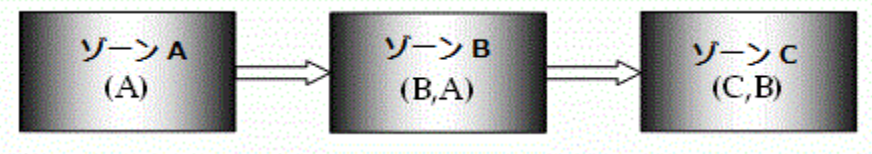
検証が正常に行われた場合は、Web エージェントはセッションの処理を停止して、正常に行われた検証からのレスポンスの処理を即時に開始します。エージェントが、検証対象のセッションを 3 つ持っている場合に、最初のセッションが正常に検証されると、残りの 2 つは無視され、エージェントは正常に行われた最初の検証に関するレスポンスの処理に移行します。

詳細情報

トラステッド ゾーンの順序 (P. 305)

ゾーン間の推移的な関係

シングル サインオン ゾーン間の信頼関係は、完全に推移的ではありません。 ゾーン A がゾーン B によって信頼され、ゾーン B がゾーン C によって信頼される場合は、以下の図に示すように、ゾーン A は必ずしもゾーン C によって信頼されるとは限りません。



シングル サインオン ゾーンの影響を受けるその他の Cookie

SiteMinder は、状態 cookie を使用して、認証と許可に関係するさまざまなイベントを管理します。 これらの cookie はすべて、デフォルトで先頭にシングル サインオン セキュリティ ゾーンのプレフィックス SM が付いています。 新しいシングル サインオン ゾーン名を指定すると、これらの cookie にも、指定されたデフォルト以外のゾーン名を反映するように名前が付けられます。 新しいシングル サインオン ゾーンを定義することで影響を受ける cookie のリストを以下に示します。

- SMCHALLENGE
- SMDATA
- SMIDENTITY
- SMONDENIEDREDIR
- SMSESSION
- SMTRYNO

たとえば、ゾーン名 Z1 を指定すると、Web エージェントは、基本認証を行うために Z1CHALLENGE=YES cookie の作成を開始します。 これによって、管理者は、エージェントが相互に干渉しない単一の cookie ドメイン（たとえば、ca.com）で SiteMinder アプリケーションのシングル サインオンの島を作成できるようになります。 このシングル サインオンのトラステッド ゾーンのリストでは、予測可能な方法で、これらの孤立したシングル サインオン ゾーン間でシングル サインオンを行うことができます。

シングル サインオン ゾーンと許可

シングル サインオン ゾーンを使用すると、変更を行わずに認証を正常に行った後で、通常は次に許可が行われます。 検証プロセスで有効なセッションが識別されると、残りの要求処理ではこのセッションが使用され、要求で識別されたその他のセッションはすべて無視されます。 許可が失敗すると、正常に許可される可能性があるセッションがほかにあるかどうかに関係なく、ユーザは再認証を要求されます。

検証に合格した最初のトラステッド セッションは、許可に渡されるセッションです。 このセッションの許可が失敗すると、ユーザは認証情報を要求されます。

セキュリティゾーンの設定

2 つのシングル サインオン パラメータが、ポリシー ストア内の Web エージェント設定オブジェクトに追加されました。 これらの設定は、ローカル設定ファイルでも使用される可能性があり、インストール時に作成されるサンプルのローカル設定ファイルに追加されます。

注: 同一のエージェント設定オブジェクトを使用して設定される Web エージェントはすべて、同一のシングル サインオン ゾーンに属します。

SSOZoneName

Web エージェントがサポートするシングル サインオン セキュリティー ゾーンの（大文字と小文字を区別する）名前を指定します。 このパラメータの値は Web エージェントが作成する cookie の名前に先頭に付けられます。 この設定は、それぞれの Cookie ドメインと Cookie を関連付けるのに役立ちます。 このパラメータが空でない場合、SiteMinder は以下の規則を使用して、cookie を生成します。

ZonenameCookiename

デフォルト： 空（ゾーン名として SM を使用します。これは Cookie に以下のデフォルトの名前を与えます）。

- SMSESSION
- SMIDENTITY
- SMDATA
- SMTRYNO
- SMCHALLENGE
- SMONDENIEDREDIR

制限： 単一値。 このパラメータは、英語の文字のみをサポートしています。-

例： Z1 に値を設定すると以下の cookie が作成されます。

- Z1SESSION
- Z1IDENTITY
- Z1DATA
- Z1TRYNO
- Z1CHALLENGE

- Z1ONDENIEDREDIR

SSOTrustedZone

シングル サインオン セキュリティ ゾーンの信頼の信頼された SSOZoneNames の順序づけられた（大文字と小文字を区別する）リストを定義します。必要に応じて、SM を使用してデフォルト ゾーンを追加します。エージェントは常に、その他すべてのトラステッド シングル サインオン ゾーンより、自身の SSOZoneName を信頼します。

デフォルト： 空（提供されている場合は SM または SSOZoneName）

制限： 複数値

エージェントのシングル サインオン ゾーンの指定

SSOZoneName

Web エージェントがサポートするシングル サインオン セキュリティー ゾーンの（大文字と小文字を区別する）名前を指定します。このパラメータの値は Web エージェントが作成する cookie の名前に先頭に付けられます。 この設定は、それぞれの Cookie ドメインと Cookie を関連付けるのに役立ちます。 このパラメータが空でない場合、SiteMinder は以下の規則を使用して、cookie を生成します。

ZonenameCookiename

デフォルト： 空（ゾーン名として SM を使用します。これは Cookie に以下のデフォルトの名前を与えます）。

- SMSESSION
- SMIDENTITY
- SMDATA
- SMTRYNO
- SMCHALLENGE
- SMONDENIEDREDIR

制限： 単一値。 このパラメータは、英語の文字のみをサポートしています。-

例： Z1 に値を設定すると以下の cookie が作成されます。

- Z1SESSION
- Z1IDENTITY
- Z1DATA
- Z1TRYNO
- Z1CHALLENGE
- Z1ONDENIEDREDIR

Web エージェントがサポートするシングル サインオン ゾーンの名前を入力するには、SSOZoneName パラメータを使用します。 このパラメータでは、大文字と小文字が区別されます。 指定されていない場合は、デフォルトで SM に設定されます。 SSOZoneName パラメータの値が空ではない場合は、Web エージェントは、以下の命名規則を使用して cookie を生成します。

zone_namecookie_name

ここで、*zone_name* はパラメータ値で、*cookie_name* は作成される cookie の一般名です。

この規則の影響を受ける Cookie には、以下のものがあります。

- SESSION
- IDENTITY
- DATA
- TRYNO
- CHALLENGE
- ONDENIEDREDIR

ユーザが、まだセッションを確立していないシングル サインオン ゾーンで検証される場合は、ポリシー サーバによって返されるセッションの指定が、そのゾーンの新規セッション cookie の作成に使用されます。

新しい cookie が作成されると、ユーザが、単に名前変更することによって別のゾーンからの cookie を交換できないようにするために、そのゾーン パラメータはゾーン名に設定されます。 cookie の検証エンジンは、ゾーン名が cookie の名前で使用されているプレフィックスと一致するかどうかを検証します。 これが適用されるのは、SESSION cookie と IDENTITY cookie のみです。

Web エージェントのサポート対象のシングル サインオン ゾーンの名前を指定するには、SSOZoneName パラメータにゾーンの名前を追加します。

信頼とフェールオーバの順序

シングル サインオン ゾーンの信頼順序を指定するには、SSOTrustedZone パラメータを使用します。 要求の処理時に、Web エージェントは、リストに表示される順に、各ゾーンの SESSION cookie または IDENTITY cookie を検索します。

見つかった cookie はすべて通常どおり検証されます（暗号化解除され、有効なホスト名、シングル サインオン ゾーン名、およびタイムアウトが指定されているかどうかがテストされます）。次に、有効な場合は、トラステッド セッションの番号付きリストに格納されます。 認証される前は、ユーザのアクティブ セッション（およびユーザ ID）は、有効なセッションの番号付きリストにおける最初のセッションであると見なされます。

認証中に、Web エージェントはリスト内の最初のセッションを使用して、検証を呼び出します。 検証が成功した場合、エージェントは先へ進んでユーザ ID を確立し、アクティブな ID として確認します。 検証が失敗した場合、新たな検証呼び出しで以下のセッションが使用されます。検証が成功するか、エージェントが指定のセッション数を使い果たすまで、同じことが繰り返されます。 どのセッションでも検証されていない場合、エージェントは通常どおりユーザに認証情報を要求します。

いったん検証されると、エージェントは他のすべてのセッションを無視して、検証済みのセッションのみに対する要求処理の残り部分について処理を進めます。 つまり、認証が失敗した場合、ユーザは直ちに認証情報を要求されることになります。 要求内の他のすべての既存セッションは使用されません。

第 17 章: 高度な構成設定

エージェントとプロキシ サーバ

プロキシ サーバ上で実行される SiteMinder エージェントを管理するために、以下の設定項目のうち必要なものを設定します。

- プロキシ サーバの背後にあるエージェントの設定
- Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ
- プロキシ ヘッダの使用に関する注意事項
- セキュリティの考慮事項

プロキシ サーバの背後にあるエージェントの設定

Web エージェントをプロキシ サーバの背後にインストールする場合、以下のパラメータを使用して、プロキシ サーバで動作する Web エージェントを設定できます。

ProxyTrust

送信先サーバ上のエージェントに対し、プロキシ サーバ上の SiteMinder エージェントから受信した許可を信頼するようにエージェントに命じます。送信先サーバはリバース プロキシ サーバの背後に置かれたサーバです。 この値を yes に設定すると、プロキシ サーバ上のエージェントのみが認可のためにポリシー サーバに問い合わせるため、効率が上がります。 送信先サーバ上で動作するエージェントは、ユーザに対する許可を再度求めるためのポリシー サーバ問い合わせを *しません*。

デフォルト： No

ExpireForProxy

クライアントがコンテンツ（ページおよび潜在的なヘッダまたは cookie）をキャッシュしないようにします。このパラメータの値が Yes に設定された場合、Web エージェントは以下の HTTP ヘッダのいずれかを HTTP レスポンスに挿入します。

- Expires
- Cache-control

コンテンツをキャッシュしない場合、後続の要求は引き続き転送されます。

ExpireForProxy パラメータが yes に設定された場合、Web エージェントでは、適切な ProxyHeaders<*suffix_name*> パラメータに指定された文字列を、エージェントが実行したリクエストの種類に基づいて、HTTP レスポンスに挿入します。

HTTP/1.1 リクエストの場合、エージェントは、以下のパラメータの値をヘッダとしてレスポンスに挿入します。

- ProxyHeadersAutoAuth
- ProxyHeadersProtected
- ProxyHeadersUnprotected

HTTP/1.0 リクエストの場合、エージェントは、以下のパラメータの値をヘッダとしてレスポンスに挿入します。

- ProxyHeadersAutoAuth10

- ProxyHeadersProtected10
- ProxyHeadersUnprotected10

デフォルト： No

注: このパラメータ名には 'proxy' という単語が含まれていますが、このパラメータの設定は、Web ブラウザ、またはこのパラメータ設定を使用する SiteMinder エージェントが動作する Web サーバに接続するすべてのクライアントの動作にも影響があります。

ページをキャッシュしないようプロキシに指示するために、Web エージェントはそのページに関する Expires ヘッダを追加します。 このヘッダは、過去の日付に設定されています。そのため、HTTP 1.0 仕様で規定されているように、プロキシがそのページをキャッシュすることは防止されます。302 リダイレクトが発生した場合、cache-control: no-cache（キャッシュ制御: キャッシュがありません）ヘッダが代わりに設定されます。 これは、コンテンツのキャッシングを防止しますが、マイクロソフト サポートによる説明のとおり、Internet Explorer（IE）ブラウザを使用している場合は、ブラウズ操作に悪影響を及ぼします。

302 リダイレクトに対して cache-control: no-cache を使用する場合、IE の中でインプレース文書の表示を管理する ActiveX コンポーネントは、ファイルを検索する際に、ブラウザのキャッシュを必要とします。 このヘッダは、ファイルをキャッシュしないようブラウザに指示するので、この ActiveX コンポーネントはファイルを検索することができず、リクエストを正しく表示することに失敗します。 さらに、Web エージェントの ExpireForProxy 設定項目を yes に設定すると、バックエンド サーバはプロキシに対し、リソースをキャッシュしないよう指示します。

プロキシ サーバの背後にあるエージェントを設定する方法

1. ProxyTust パラメータに yes を設定します。
2. ExpireForProxy パラメータに yes を設定します。
3. （任意）Cache-Control および ExpireForProxy（HTTP）ヘッダの値をカスタマイズします。

   プロキシ サーバの背後にあるエージェントが設定されます。

詳細情報:

Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ (P. 318)

Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ

cache-control と ExpireForProxy の各ヘッダをカスタマイズして、アプリケーション ファイル（.doc、.pdf など）のインプレース起動に影響を及ぼすことなく、Web リソースを安全にすることができます。 以下のコンテンツ タイプの特定の HTTP ヘッダを別途設定することによって、Web ブラウザまたはプロキシ サーバによってコンテンツがどのようにキャッシュされるかを制御することができます。

- 自動許可されている
- 保護されていない
- 保護されている

重要: RFC 2068 に準拠したこれらの設定の変更の影響がよくわからない場合は、デフォルトの設定を使用することをお勧めします。 デフォルトの設定を変更する予定がある場合は、ユーザがアイドル タイムアウトを追跡するためにセッションを確立した後で、保護されていないページにアクセスすると、SiteMinder セッション cookie が更新されることに注意してください。したがって、保護されていないページを、HTTP ヘッダをキャッシュするプロキシのキャッシュ対象とすることは望ましくありません。

プロキシによるキャッシュを防ぐために、以下の特性が設定ヘッダに適用されます。

- エージェントのアクティビティにかかわらず、すべてのリダイレクトで Cache-Control: no-cache ヘッダが設定されます。
- Web サーバは、使用されている HTTP プロトコル（1.0 または 1.1 以上）に基づいて、プロキシ/クライアントに適切なヘッダを送り返します。

cache-control: private や cache-control: max-age=60 のような複数のヘッダを使用することに対応して、あらゆるパラメータは、複数の値を表す文字列を使用して設定する必要があります。

次に、新しい設定について説明します。

1. ProxyHeadersDefaultTime - デフォルトは 60 秒です。
2. ProxyHeadersTimeoutPercentage - デフォルトは 10% です。
3. 以下の Cache-Control ヘッダを使用できます。

   ProxyHeadersAutoAuth

Web エージェント設定の ExpireForProxy パラメータが yes に設定されている場合、Web エージェントがクライアントへの HTTP レスポンスに挿入する HTTP 1.1 ヘッダの値を指定します。 このヘッダの値によって、自動許可されたリソースがキャッシュされるかどうか、またはキャッシュされる期間が決定します。

デフォルト： Expires: Thu, 01 Dec 1994 16:00:00 GMT

例（推奨される設定）："Cache-control: max-age=60"

ProxyHeadersAutoAuth10

Web エージェント設定の ExpireForProxy パラメータが yes に設定されている場合、Web エージェントがクライアントへの HTTP レスポンスに挿入する HTTP 1.0 ヘッダの値を指定します。 このヘッダの値によって、自動許可されたリソースがキャッシュされるかどうか、またはキャッシュされる期間が決定します。

デフォルト： Expires: Thu, 01 Dec 1994 16:00:00 GMT

例（推奨される設定）："Expires: Thu, 01 Dec 1994 16:00:00 GMT"

ProxyHeadersProtected

Web エージェント設定の ExpireForProxy パラメータが yes に設定されている場合、Web エージェントがクライアントへの HTTP レスポンスに挿入する HTTP 1.1 ヘッダの値を指定します。 このヘッダの値によって、保護されているリソースがキャッシュされるかどうか、またはキャッシュされる期間が決定します。

デフォルト： Expires: Thu, 01 Dec 1994 16:00:00 GMT

Cache-Control: no-cache

例（推奨される設定）："Cache-Control: private"

ProxyHeadersProtected="Cache-Control: max-age=60"

ProxyHeadersProtected10

Web エージェント設定の ExpireForProxy パラメータが yes に設定されている場合、Web エージェントがクライアントへの HTTP レスポンスに挿入する HTTP 1.0 ヘッダの値を指定します。 このヘッダの値によって、保護されているリソースがキャッシュされるかどうか、またはキャッシュされる期間が決定します。

デフォルト： Expires: Thu, 01 Dec 1994 16:00:00 GMT

Cache-Control: no-cache

例（推奨される設定）："Expires: Thu, 01 Dec 1994 16:00:00 GMT"

ProxyHeadersUnprotected

Web エージェント設定の ExpireForProxy パラメータが yes に設定されている場合、Web エージェントがクライアントへの HTTP レスポンスに挿入する HTTP 1.1 ヘッダの値を指定します。 このヘッダの値によって、保護されていないリソースがキャッシュされるかどうか、またはキャッシュされる期間が決定します。

デフォルト： Expires: Thu, 01 Dec 1994 16:00:00 GMT

Cache-Control: no-cache

例（推奨される設定）： ProxyHeadersUnprotected="Cache-Control: private"

ProxyHeadersUnprotected="Cache-Control: max-age=60"

ProxyHeadersUnprotected10

Web エージェント設定の ExpireForProxy パラメータが yes に設定されている場合、Web エージェントがクライアントへの HTTP レスポンスに挿入する HTTP 1.0 ヘッダの値を指定します。 このヘッダの値によって、保護されていないリソースがキャッシュされるかどうか、またはキャッシュされる期間が決定します。

デフォルト： Expires: Thu, 01 Dec 1994 16:00:00 GMT

Cache-Control: no-cache

例（推奨される設定）： "Expires: Thu, 01 Dec 1994 16:00:00 GMT"

複数のヘッダを設定する場合（たとえば、保護されていない HTTP/1.1 コンテンツに対して、cache-control ヘッダを推奨設定値にする場合）、以下のことに注意してください。

- 設定パラメータを複数の箇所で記述する*必要があります*。また、それらの値をカンマ（,）またはプラス記号（+）で区切ることはできません。
- これらの設定パラメータの値は HTTP レスポンス ヘッダの値なので、RFC 2616（HTTP/1.1 用）、RFC 1945（HTTP/1.0 用）、および RFC 822 に準拠する必要があります。 HTTP/1.1 と HTTP/1.0 の両方が HTTP ヘッダの形式を RFC 822 メッセージの形式として指定します。つまり、「Name: Value」となります（Name の後にコロン、スペース、値が続きます）。

保護されていないリソースにユーザがアクセスした場合に、適切なキャッシュ期限ヘッダを設定するよう Web エージェントが設定されていない場合、Web エージェントはデフォルトではそれらのヘッダを設定しません。したがって、Web ブラウザまたはプロキシ サーバが SMSESSION cookie をキャッシュすることが許可されます。 このキャッシュされた cookie は、ユーザが他のセッション（別のユーザ コンテキスト）を開始すると、Web ブラウザまたはプロキシ サーバによって再利用されるため、許可されていないインパーソネーション（偽装）が発生します。

詳細情報:

プロキシ サーバの背後にあるエージェントの設定 (P. 316)

プロキシ ヘッダの使用に関する注意事項

- Web エージェントがどのプロキシ ヘッダも送信しないように設定するには、ProxyHeadersUnprotected の値を空白にします。 例：

  ProxyHeadersUnprotected=""

  注: 二重引用符 (") を表示するには、引用符 (') を使用します。Web エージェントは、引用符を自動的に二重引用符へ変換します。

- %% または %d という値（同じものとして取り扱われます）は、ProxyHeaders 行の中で記述することができます。この値は、IdleTimeout および SessionTimeout のうち小さいものに対して、ProxyHeadersTimeoutPercentage をかけた値へ置き換えられます。タイムアウトが設定されていない場合、ProxyHeadersDefaultTime が使用されます。
- 標準的なヘッダ（1.1 およびそれ以降）と、HTTP 1.0 ヘッダーのそれぞれに関して、バックエンド サーバへのリクエストを想定して、これらの値が正しく設定されていることを確認してください。
- ExpireForProxy="YES" は、クエリ 文字列の中で SMSESSION cookie を渡す、cookie プロバイダによるリダイレクトを期限切れにします。

セキュリティの考慮事項

ブラウザセッションは、ログアウトの後も持続します。そのため、SMSESSION cookie を削除した場合も、ユーザが同じブラウザセッションを使用して、既にキャッシュされたファイルを表示する作業を防止することはありません。 この問題が発生する原因は、以下のとおりです。プロキシ サーバはログアウト要求を意識せず、保護されたコンテンツ/保護されていないコンテンツのすべてが、タイムアウトになる（cache-control: max-age=60）までは、cache-control: private ユーザ用にそれらをキャッシュの中にとどめます。 したがって、1 つのページ内でそのようなリクエストを実行した場合、有効な SMSESSION cookie が返されます。 セキュリティを保証する唯一の方法は、キープ アライブを無効にすること、またはそのブラウザを閉じることです。

さらに、ローカルブラウザキャッシュは、private/max-age の組み合わせから影響を受けます。そのキャッシュは、複数のセッションにわたってローカルキャッシュを参照するからです。 この理由で、保護されたリソースに関連する max-age の時間は、できるだけ短くする必要があります。

allowcacheheaders="FALSE" 設定ヘッダが使用されている（デフォルト）状況で、if-modified-since と if-none-match の各要求ヘッダを利用する場合、プロキシ サーバがこれらのヘッダを参照することは妨げられません。 したがって、プロキシ サーバによるリクエストに対して、これらの参照されたヘッダが有効になります。

以下をインストールすることにより、この課題を回避することもできます。

- プロキシ サーバ上に Web エージェント。
- リクエストからこれらのヘッダを削除する他のフィルタ。

HTTP 1.0 と、HTTP 1.1 またはそれ以降は、キャッシングを行うプロキシに対して命令を指定する目的で、互いに異なるヘッダを使用します。そのため、接続のタイプに基づいて最適な取り扱いを保証するために、これらのバージョンを 1 つの方法で設定する必要があります。

エージェントおよびリバース プロキシ サーバ

リバース プロキシ サーバ上で展開する SiteMinder エージェントを管理するには、以下のトピックを参照してください。

- SiteMinder でのリバース プロキシ サーバの機能
- SiteMinder セキュア プロキシ サーバ用 SM_PROXYREQUEST HTTP ヘッダの設定
- IIS Web サーバへのアプリケーション リクエスト ルーティング（ARR）の実装
- リバース プロキシ展開の要因について（P. 335）
- リバース プロキシとして Apache ベースの Web サーバを設定する
- リバース プロキシとして Oracle iPlanet 7.0 サーバを設定する

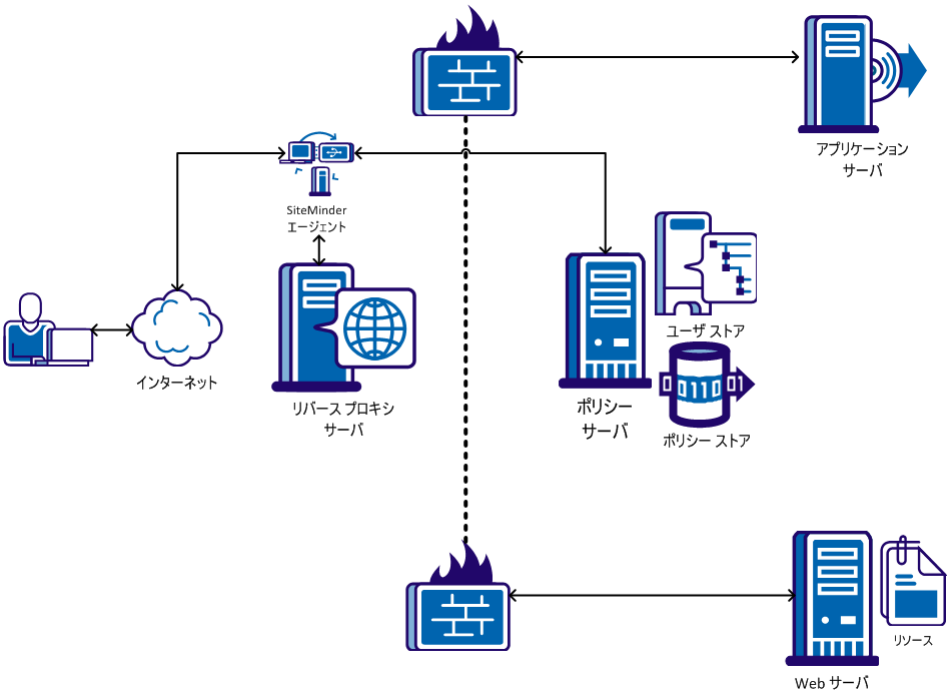
SiteMinder でのリバース プロキシ サーバの機能

リバース プロキシ サーバは、企業の代理として、組織の内部ネットワークに要求を転送する機能を持つプロキシ サーバです。 リバース プロキシ サーバは、バックエンド サーバ（ファイアウォールの背後にあるサーバ）上のリソースに、クライアントがアクセスすることを可能にします。

リバース プロキシ サーバを使用すると次の利点があります。

- Cookie ドメイン内のユーザは、再認証しなくてもバックエンド サーバ上のリソースにアクセスできます。 他のドメインのユーザは、リバース プロキシ サーバの認証（通常はファイアウォールの認証も）を受けないと、それらの同じバックエンド サーバにアクセスできません。
- ユーザは、同じドメイン名を使用して、いくつかのバックエンド サーバ上でホストされている別のリソースにアクセスできます。
- リバース プロキシ エージェントは他の SiteMinder エージェントと同じ機能をサポートします。
- SiteMinder エージェントがサポートされていないサーバ上にあるリソースの保護。この状況では、バックエンド サーバの前にリバース プロキシ サーバを展開します。サポートされるエージェントは、バックエンド サーバ上でホストされたリソースを保護します。バックエンド サーバは SiteMinder エージェントを必要としません。

リバース プロキシ サーバにインストールされる SiteMinder エージェントは、バックエンド サーバ上のリソースを保護できます。 次の図に、SiteMinder エージェントを使用するリバース プロキシ サーバを使用したネットワークを示します。



SiteMinder セキュア プロキシサーバ

より高度なリバース プロキシ ソリューションを必要とする場合は、CA SiteMinder for Secure Proxy Server を使用できます。これには、Apache または Oracle iPlanet ベースの SiteMinder リバース プロキシ エージェントに比べて以下のような利点があります。

- Web サーバが組み込まれ完全にサポートされています。これには、SSL アクセラレータ カードのサポートや、キーと証明書の管理を行う GUI ツールなどが含まれます。
- 複数のセッション方式のサポート（cookie ベースと cookie なし）
- 以下のような、柔軟なプロキシ ルールのサポート。
  - URL に加えて、HTTP ヘッダと SiteMinder レスポンスに基づくルールをサポートしています。
  - 複雑なルールを簡単に取り扱うことができます。

セキュア プロキシ サーバによる SiteMinder 処理用の SM_PROXYREQUEST HTTP ヘッダ

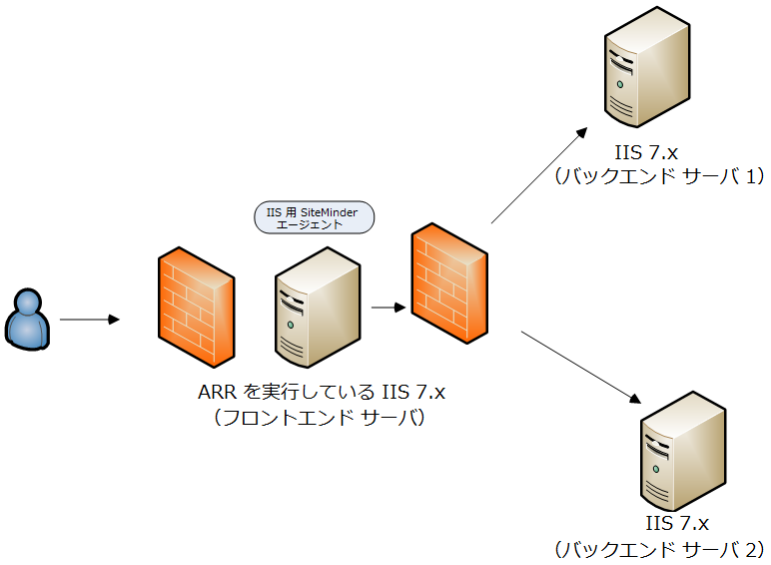
CA SiteMinder for Secure Proxy Server では、従来の SiteMinder アーキテクチャに新しいレイヤが導入されます。 このレイヤはすべての要求を企業内の宛先サーバに転送またはリダイレクトします。

CA SiteMinder for Secure Proxy Server が要求を処理する際、ユーザが要求した URL は SM_PROXYREQUEST という HTTP ヘッダ変数内に保持されます。 CA SiteMinder for Secure Proxy Server が要求をプロキシする前の、ユーザが要求したオリジナルの URL を必要とする他のアプリケーションは、このヘッダを使用できます。

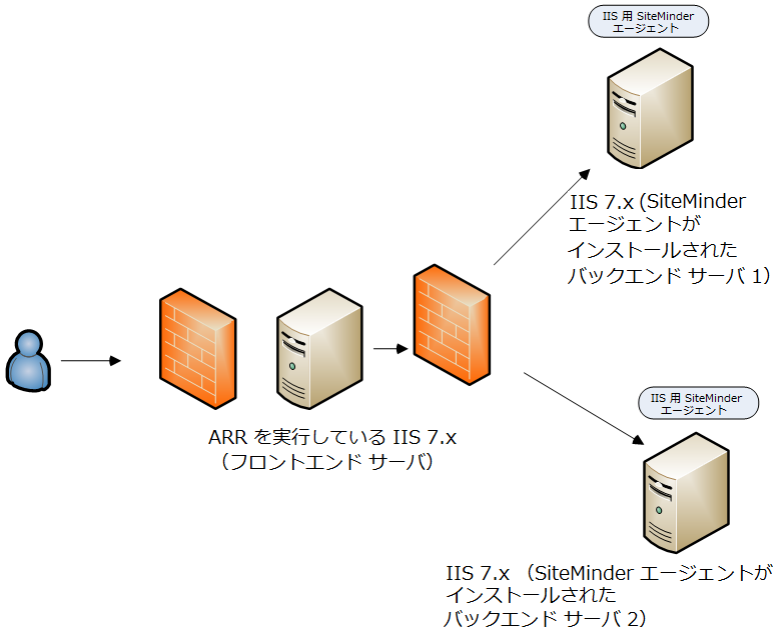
SiteMinder IIS 7.x Web サーバおよびアプリケーション要求ルーティング（ARR）

IIS 用の SiteMinder 12.52 エージェントは、IIS 7.x のアプリケーション要求ルーティング機能をサポートします。 以下の構成がサポートされます。

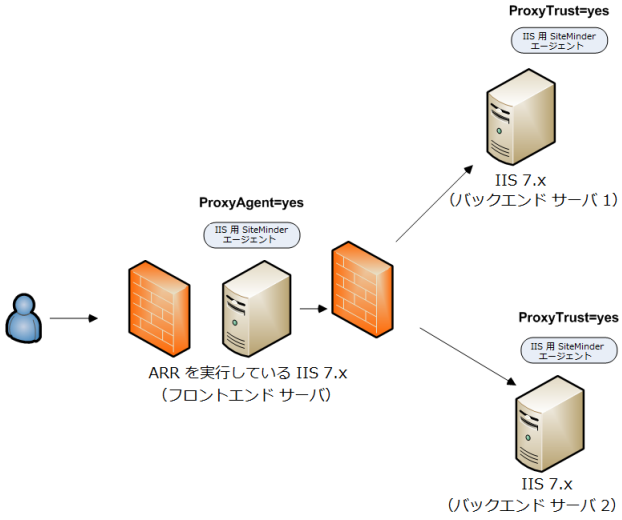
- 以下の図で示される、ARR、および DMZ 内の IIS 用の SiteMinder エージェントの両方を実行する IIS 7.x Web サーバ：（P. 333）



- ARR を実行する DMZ 内の別の IIS 7.x サーバの後ろの IIS 用の SiteMinder Web エージェントを実行する複数の IIS 7.x Web サーバ。 この構成を以下の図に示します。 (P. 334)



- ARR、および DMZ 内の IIS 用の SiteMinder エージェントの両方を実行する IIS 7.x Web サーバ、および ARR サーバの後ろの IIS 用の SiteMinder エージェントを実行する複数の IIS 7.x Web サーバ。 この構成を以下の図に示します。 (P. 328)



DMZ 内に ARR と SiteMinder で IIS 7.x サーバをセットアップし、DMZ の背後で動作する他の IIS 用 SiteMinder エージェントをセットアップする方法

IIS 用の SiteMinder エージェントは、以下の設定を使用するユーザの IIS 環境全体を保護します。

- DMZ（フロントエンド サーバとして）でアプリケーション要求ルーティング（ARR）と IIS 用の SiteMinder エージェントを使用した IIS 7.x Web サーバ。
- DMZ 内の ARR サーバの*背後*の複数の IIS 7.x Web サーバで、*それぞれ*が SiteMinder Web エージェント*または* IIS 用エージェントを使用。

  注: リバース プロキシ サーバとしての作動をサポートしている SiteMinder Web エージェントは、一部のみです。 ただしサポート対象の SiteMinder Web エージェントまたは IIS 用エージェントをホストしているすべての Web サーバは、SiteMinder を実行しているリバース プロキシ サーバからのトラフィックを受け入れることができます。詳細については、「プラットフォーム サポート マトリックス」を参照してください。

前の設定を実装するには、以下の複数手順のプロセスを使用します。

1. ARR を DMZ（フロントエンド）内の IIS 7.x Web サーバにインストールし設定します。

   注: アプリケーション要求ルーティング（ARR）の詳細については、IIS Web サイトに移動し、語句「アプリケーション要求ルーティング」を検索してください。

2. DMZ（フロントエンド）内の IIS 7.x Web サーバ上の IIS 用の SiteMinder エージェントをインストールし設定します。

   注: 詳細については、「IIS 用 Web エージェント インストール ガイド」を参照してください。

3. DMZ で IIS 用の SiteMinder エージェント用の Web エージェント設定パラメータを設定します。（P. 330）

4. DMZ（バックエンド）の*後ろ*の最初の IIS 7.x Web サーバ上の IIS 用の SiteMinder エージェントをインストールし設定します。 詳細については、「IIS 用 Web エージェント インストール ガイド」を参照してください。

注: このコンテキストでは、最初のサーバとは、共有設定情報が格納されているファームの IIS Web サーバを指します。 ノードとは、最初のサーバから共有設定を読み取った、ファームの他の IIS Web サーバのことです。

5. DMZ（バックエンド）の*後ろの*他の IIS 7.x Web サーバ ノード上の IIS 用の SiteMinder エージェントをインストールし設定します。
6. DMZ の*後ろの* SiteMinder を使用して、IIS 7.x サーバのすべて用の Web エージェント設定パラメータを設定します。（P. 332）*最初の* Web サーバと すべてのノードを含みます。

DMZ で IIS 7.x ARR サーバ用の SiteMinderWeb エージェント設定パラメータを設定する

このセクションでは、次の状況において IIS 用 SiteMinder エージェントを実行する Web エージェント設定パラメータを設定する方法について説明します。

- IIS 7.x Web サーバは、ARR および IIS 用 SiteMinder エージェントを使用して DMZ で作動します。（フロントエンド）
- DMZ の背後に置かれた他の IIS 7.x Web サーバは ARR サーバから要求を受信しますが、IIS 用 SiteMinder エージェントを使用 *しません*。（バックエンド）

次の手順に従ってください:

1. 以下の項目を確認します。
   - DMZ 内の Web サーバに ARR 2.0 がインストールされ設定されている。
   - DMZ 内の Web サーバに IIS 用 SiteMinder 12.52 エージェントがインストールされ設定されている。
2. 管理 UI を開きます。
3. IIS 用 SiteMinder エージェント（DMZ で実行されているフロントエンド）に「関連付けられている、エージェント設定オブジェクト（ACO）を開きます。
4. 以下のパラメータを探します。

   ProxyTrust

   送信先サーバ上のエージェントに対し、プロキシ サーバ上の SiteMinder エージェントから受信した許可を信頼するようにエージェントに命じます。送信先サーバはリバース プロキシ サーバの背後に置かれたサーバです。 この値を yes に設定すると、プロキシ サーバ上のエージェントのみが認可のためにポリシー サーバに問い合わせるため、効率が上がります。 送信先サーバ上で動作するエージェントは、ユーザに対する許可を再度求めるためのポリシー サーバ問い合わせを *しません*。

   デフォルト： No
5. ProxyTrust パラメータに設定された値が no であることを確認します。
6. 以下のパラメータを探します。

   ProxyAgent

Web エージェントがリバース プロキシ エージェントとして動作するかどうかを指定します。

このパラメータの値が yes である場合、フロントエンド サーバ上の SiteMinder エージェントはユーザによって要求された SM_PROXYREQUEST HTTP ヘッダ内の元の URL を維持します。 保護されているリソースと保護されていないリソースが要求された場合は常に、このヘッダが作成されます。 バックエンド サーバは、元の URL に関する情報を取得するためにこのヘッダを読み取ることができます。

デフォルト： No

7. ProxyAgent パラメータの値を yes に変更します。
8. エージェント設定オブジェクトへの変更をサブミットします。

   Web エージェント設定パラメータが設定されます。

DMZ の背後に置かれた SiteMinder を使用する IIS 7.x サーバ用の Web エージェント設定パラメータの設定

このセクションでは、次の状況において IIS 用 SiteMinder エージェントを実行する Web エージェント設定パラメータを設定する方法について説明します。

- IIS 7.x サーバは ARR を使用して、DMZ で動作します。（フロント エンド）
- DMZ の背後に置かれた他の IIS 7.x サーバは ARR サーバから要求を受信します。それらのサーバも、IIS 用 SiteMinder エージェントを使用します。（バックエンド）

次の手順に従ってください:

1. 以下の項目を確認します。
   - DMZ 内の Web サーバに ARR 2.0 がインストールされ設定されている。
   - DMZ の *背後*に置かれた最初の Web サーバおよびすべてのノードに、IIS 用 SiteMinder 12.52 エージェントがインストールされ設定されている。
2. 管理 UI を開きます。
3. DMZ の *内部*に展開されている最初の IIS サーバに関連付けられた、エージェント設定オブジェクト（ACO）を開きます。
4. 以下のパラメータを探します。

   ProxyTrust

   送信先サーバ上のエージェントに対し、プロキシ サーバ上の SiteMinder エージェントから受信した許可を信頼するようにエージェントに命じます。送信先サーバはリバース プロキシ サーバの背後に置かれたサーバです。 この値を yes に設定すると、プロキシ サーバ上のエージェントのみが認可のためにポリシー サーバに問い合わせるため、効率が上がります。 送信先サーバ上で動作するエージェントは、ユーザに対する許可を再度求めるためのポリシー サーバ問い合わせを *しません*。

   デフォルト： No
5. ProxyTrust パラメータの値を yes に変更します。
6. 以下のパラメータを探します。

   ProxyAgent

Web エージェントがリバース プロキシ エージェントとして動作するかどうかを指定します。

このパラメータの値が yes である場合、フロントエンド サーバ上の SiteMinder エージェントはユーザによって要求された SM_PROXYREQUEST HTTP ヘッダ内の元の URL を維持します。 保護されているリソースと保護されていないリソースが要求された場合は常に、このヘッダが作成されます。 バックエンド サーバは、元の URL に関する情報を取得するためにこのヘッダを読み取ることができます。

デフォルト： No

7. ProxyAgent パラメータの値が no に設定されていることを確認します。
8. エージェント設定オブジェクトへの変更をサブミットします。
9. DMZ の*内部*に展開された IIS サーバ ノードに関連付けられた、エージェント設定オブジェクト（ACO）を開きます。
10. DMZ 内部のすべてのノードが設定されるまで、各 IIS Web サーバ ノード上で手順 5 ～ 10 を繰り返します。

    Web エージェント設定パラメータが設定されます。

DMZ 内に ARR と SiteMinder で IIS 7.x サーバをセットアップする方法

ユーザの DMZ（フロントエンド サーバとして）でアプリケーション要求ルーティング（ARR）を使用して IIS 7.x Web サーバおよび IIS 用の SiteMinder エージェントをセットアップするには、以下の複数手順のプロセスを使用します。

1. ARR を DMZ（フロントエンド）内の IIS 7.x Web サーバにインストールし設定します。

   注: アプリケーション要求ルーティング（ARR）の詳細については、IIS Web サイトに移動し、語句「アプリケーション要求ルーティング」を検索してください。

2. DMZ（フロントエンド）内の IIS 7.x Web サーバ上の IIS 用の SiteMinder エージェントをインストールし設定します。

   注: 詳細については、「IIS 用 Web エージェント インストール ガイド」を参照してください。

DMZ 内の ARR サーバの背後で動作しているときに SiteMinder で IIS 7.x サーバをセットアップする方法

IIS 用の SiteMinder エージェントは、アプリケーション リクエスト ルーティング（ARR）を使用する以下の設定をサポートします。

- ARR を実行する DMZ ベースの IIS 7.x Web サーバの *背後* で動作する複数のバックエンド Web サーバ。
- SiteMinder Web エージェントまたは IIS 用エージェントによるこれらのバックエンド サーバの保護。

  注: リバース プロキシ サーバとしての作動をサポートしている SiteMinder Web エージェントは、一部のみです。 ただしサポート対象の SiteMinder Web エージェントまたは IIS 用エージェントをホストしているすべての Web サーバは、SiteMinder を実行しているリバース プロキシ サーバからのトラフィックを受け入れることができます。詳細については、「プラットフォーム サポート マトリックス」を参照してください。

この設定を実装するには、以下の複数手順のプロセスを使用します。

1. ARR を DMZ（フロントエンド）内の IIS 7.x Web サーバにインストールし設定します。

   注: アプリケーション要求ルーティング（ARR）の詳細については、IIS Web サイトに移動し、語句「アプリケーション要求ルーティング」を検索してください。

2. DMZ（バックエンド）の *後ろの* 最初の IIS 7.x Web サーバ上の IIS 用の SiteMinder エージェントをインストールし設定します。 詳細については、「IIS 用 Web エージェント インストール ガイド」を参照してください。

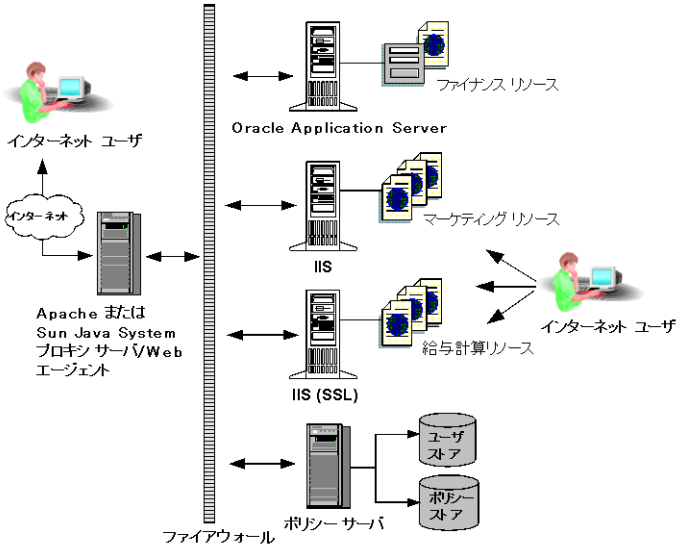
   注: このコンテキストでは、最初のサーバとは、共有設定情報が格納されているファームの IIS Web サーバを指します。 ノードとは、最初のサーバから共有設定を読み取った、ファームの他の IIS Web サーバのことです。

3. DMZ（バックエンド）の *後ろの* 他の IIS 7.x Web サーバ ノード上の IIS 用の SiteMinder エージェントをインストールし設定します。

SiteMinder リバース プロキシ展開の考慮事項

通常は、Apache または Oracle iPlanet リバース プロキシ エージェントを展開する場合、Apache または Oracle iPlanet Web エージェントと保護対象リソースをホストするサーバとの間にファイアウォールが存在します。 また、ポリシー サーバもファイアウォールの背後に配置する必要があります。

以下の図に、SiteMinder リバース プロキシの展開を示します。



SiteMinder リバース プロキシ エージェントを展開する際は、以下の点を考慮してください。

- ポリシーがレスポンス属性を返すように設定されている場合、それらのレスポンス変数は、リバース プロキシ サーバと保護対象リソースが存在するバックエンドの Web サーバの両方に送信されます。 保護対象リソースへの要求が発生すると、ポリシー サーバはまず、レスポンス属性（CGI または HTTP 変数）を Apache または Oracle iPlanet サーバ上のエージェントに送信します。 次に、エージェントは受信したレスポンス属性をリクエスト内に挿入し、バックエンド サーバに送信します。
- バックエンド サーバや保護対象アプリケーションに独自の認証機能が備わっている場合、それらの認証機能は無効にしておく必要があります。 バックエンド認証を無効にすると、SiteMinder の認証が優先されるようになります。

  重要: リバース プロキシのキャッシュを設定すると、SMSESSION cookie を含め、すべての cookie がキャッシュに格納されます。 詳細については、Apache または Oracle iPlanet Web サーバのドキュメントを参照してください。

詳細情報

HTTPS ポートの定義 (P. 116)

Apache リバース プロキシ サーバを設定する方法

Apache Web サーバを、任意の SiteMinder エージェントを持つリバース プロキシ サーバとして機能するように設定できます。 次に、Apache リバース プロキシ サーバを設定するための手順を示します。

1. Apache Web サーバ構成ファイルを更新します (P. 337)。
2. SiteMinder エージェント用のエージェント設定パラメータを更新します (P. 339)。

Apache Web サーバ構成ファイルの更新

Apache Web サーバをリバース プロキシ サーバとして機能させるために、SiteMinder エージェントを使用して、Apache Web サーバの設定ファイルを更新します。

次の手順に従ってください:

1. 以下の場所にある httpd.conf ファイルを開きます。

   ```
   /etc/httpd/conf/httpd.conf
   ```

2. 次のディレクティブを httpd.conf ファイルに追加します。

   ProxyPass

   リモート サーバからローカル サーバへのマッピングを許可します。 このディレクティブの値は以下の形式を使用します。

   /local_virtual_path partial_URL_of_remote_server

   例： ProxyPass /realma/ http://server.example.org/realma/

   ProxyPassReverse

   HTTP リダイレクト レスポンス上での Apache サーバによるロケーション ヘッダの調整を許可します。 このディレクティブの値は以下の形式を使用します。

   /local_virtual_path partial_URL_of_remote_server

   例： ProxyPassReverse /realma/ http://server.example.org/realma/

   Apache Web サーバについては、設定ファイルへ以下のプロキシ パス設定を追加します。

```
# SiteMinder Administrative UI
<Location "/iam/siteminder/">
     <IfModule proxy_module>
                 ProxyPass http://hostname:port/iam/siteminder/
       ProxyPassReverse http://hostname:port/iam/siteminder/
    </IfModule>
  # Alternate unavailable page
  ErrorDocument 503 /siteminderagent/adminui/HTTP_SERVICE_UNAVAILABLE.html
</Location>
# CA Styles r5.1.1
<Location "/castylesr5.1.1/">
    <IfModule proxy_module>
     ProxyPass http://hostname:port/castylesr5.1.1/
    ProxyPassReverse http://hostname:port/castylesr5.1.1/
    </IfModule>
</Location>
```

注: *hostname:port* は、管理 UI を実行するアプリケーション サーバのホストおよびポートを参照します。

3. 設定ファイル内の以下の行のコメントを解除します。

   ```
   LoadModule proxy_module modules/mod_proxy.so
   ```

4. 設定ファイルを保存して閉じます。
5. Apache Web サーバを再起動します。

SiteMinder エージェントのエージェント設定パラメータの更新

Apache リバース プロキシ サーバの背後に置かれた Apache ベースのサーバについては、以下のエージェント設定パラメータを更新します。

次の手順に従ってください:

1. 以下のパラメータの値を yes に設定します。

   ProxyAgent

   Web エージェントがリバース プロキシ エージェントとして動作するかどうかを指定します。

   このパラメータの値が yes である場合、フロントエンド サーバ上の SiteMinder エージェントはユーザによって要求された SM_PROXYREQUEST HTTP ヘッダ内の元の URL を維持します。 保護されているリソースと保護されていないリソースが要求された場合は常に、このヘッダが作成されます。 バックエンド サーバは、元の URL に関する情報を取得するためにこのヘッダを読み取ることができます。

   デフォルト： No

2. 以下のパラメータを設定します。

   ProxyTimeout

   要求に応答するために、リバース プロキシ サーバーの背後に展開された SiteMinder エージェントをリバース プロキシ サーバが待機する秒数を指定します。

   デフォルト： 120

   注: このパラメータは Apache ベースのエージェントにのみ適用されます。

3. （オプション）以下のパラメータを設定します。

   ProxyTrust

   送信先サーバ上のエージェントに対し、プロキシ サーバ上の SiteMinder エージェントから受信した許可を信頼するようにエージェントに命じます。送信先サーバはリバース プロキシ サーバの背後に置かれたサーバです。 この値を yes に設定すると、プロキシ サーバ上のエージェントのみが認可のためにポリシー サーバに問い合わせるため、効率が上がります。 送信先サーバ上で動作するエージェントは、ユーザに対する許可を再度求めるためのポリシー サーバ問い合わせを *しません*。

デフォルト： No

4. リストから以下の値をすべて削除することにより、BadURLChars パラメータを編集します。

   %

5. SSL 用にセットアップされたポートを Apache サーバに示すために、httpsports パラメータを設定します。

6. Apache Web サーバを再起動します。

   注: エージェント設定パラメータ変更の詳細については、「ポリシー サーバ設定ガイド」を参照してください。

Oracle iPlanet 7.0 リバース プロキシ サーバの設定

SiteMinder で Oracle iPlanet 7.0 Web サーバをリバース プロキシとして使用することができます。

注: SiteMinder エージェント設定ウィザードは、Oracle iPlanet（以前の Sun Java System）Web サーバ上のデフォルトの obj.conf ファイルのみを変更します。SiteMinder で他のインスタンスまたはリバース プロキシ展開を保護するには、デフォルトの obj.conf ファイルから、対応する <instance_name>-obj.conf ファイルに SiteMinder 設定をコピーします。 たとえば、Web サーバのインストール時に obj.conf ファイルが作成されましたが、その後 my_server.example.com という名前のサーバ インスタンスを追加したとします。 SiteMinder で my_server.example.com 上のリソースを保護するには、obj.conf ファイルから my_server.example.com-obj.conf ファイルに、ウィザードによって追加された SiteMinder 設定をコピーします。

次の手順に従ってください:

1. 以下のディレクティブを *instance_name*-obj.conf ファイルに追加します。

   NameTrans

   以下の形式を使用して、ローカルおよびリモートの仮想パスを指定します。

   NameTrans fn="map" from="*local_virtual_path*" name="reverse-proxy-/local_virtual_path" to="*remote_virtual_path*"

   例： NameTrans fn="map" from="/realma" name="reverse-proxy-/reamla" to="http://server.example.org/realma/"

2. 以下のディレクティブを obj.conf ファイルの末尾に追加します。

   オブジェクト名

   以下の形式を使用して、NameTrans ディレクティブ内で使用されるローカルの仮想パスの名前とリモートの仮想パスの URL を指定します。

   <Object name="reverse-proxy-/*local_virtual_path*">

   Route fn="set-origin-server" server="http://*remote_server_URL*:*port*"

   </Object>

   例： <Object name="reverse-proxy-/realma">

   Route fn="set-origin-server" server="http://server.example.org:*port*"

   </Object>

Object ppath

クライアントからサーバに与えられた部分的なパスを指定します。

例： <Object ppath="http:*">

Service fn="proxy-retrieve" method="*"

</Object>

3. Web サーバを再起動します。

リバース プロキシが設定されます。

HTTP ヘッダの設定

URL 処理を制御するために、以下の設定項目のうち、必要なものを設定します。

- URLScan ユーティリティに対応するためにサーバ HTTP ヘッダを削除

URLScan ユーティリティを使用する場合のサーバ HTTP ヘッダの削除

Microsoft の URLScan ユーティリティを使用して、IIS Web サーバが送信するレスポンスからサーバ HTTP ヘッダを削除する場合は、IIS Web エージェントの以下のパラメータも設定する必要があります。

SuppressServerHeader

IIS Web エージェントがレスポンスでサーバ HTTP ヘッダを返すことを防ぎます。このパラメータの値が no の場合、Web エージェントはレスポンスと一緒にサーバ ヘッダを送信し、IIS Web サーバはそれをクライアントに渡します。 このパラメータの値が yes の場合、Web エージェントは、レスポンスでサーバ ヘッダを送信しません。

デフォルト： No

URLScan utiltiy が IIS サーバのレスポンスからヘッダを削除するのに対し、hte SuppressServerHeader パラメータは Web エージェントのレスポンスからヘッダを削除します。 すべてのレスポンスでサーバ ヘッダがクライアントに送信されないようにするには、ユーティリティとパラメータの両方を設定する必要があります。

Web エージェントがレスポンスでサーバ ヘッダを送信しないようにするには、SuppressServerHeader パラメータの値を yes に設定します。

URL 設定

URL が処理される方法を制御するために、以下の設定項目のうち、必要なものを設定します。

- 小文字を使用したプロトコルの指定
- URL 内のクエリ データのデコード
- URL の最大サイズの設定

小文字でのリダイレクト URL プロトコルの指定

RFC 2396 に準拠しないレガシー アプリケーションをフォーム ベース認証方式で保護する場合、URL のプロトコル部分を小文字にする必要があれば、以下のパラメータを設定します。

LowerCaseProtocolSpecifier

リダイレクト URL のスキーム（プロトコル）部分で小文字のみを使用するかどうかを指定します。 この設定パラメータは、RFC 2396 に準拠していないレガシー アプリケーションに対応します。 この RFC には、アプリケーションは URL のプロトコル部分で大文字と小文字の両方を処理する必要があると記載されています。 以下のいずれかの状況でこのパラメータを変更します。

- RFC 2396 に準拠していないレガシー アプリケーションを使用する場合。
- リダイレクト URL にクエリ データが含まれている場合。
- HTML フォーム（FCC）認証方式を使用する場合。

デフォルト： No （HTTP、HTTPS のように大文字を使用）

例： Yes （http、https のように小文字を使用）

お使いの環境で URL に小文字のプロトコルを指定するには、LowerCaseProtocolSpecifier パラメータの値を yes に設定します。

URL 内のクエリ データのデコード

ポリシー サーバを呼び出す前に、Web エージェントの Base64 アルゴリズムが URL のクエリ データをデコードするように設定する（それによってポリシー サーバは適切なリソースを参照します）には、以下のパラメータを使用します。

DecodeQueryData

ポリシー サーバをコールする前に、Web エージェントが URL 内のクエリ データをデコードするかどうかを指定します。 環境内で以下のタスクのいずれかを実行する必要がある場合は、このパラメータを yes に設定します。

- 正しい文字列に対してルール ファイラが機能していることを保証する必要がある場合
- クエリ文字列内のデータに対して書き込みルールが機能していることを保証する必要がある場合

デフォルト： No

ポリシー サーバをコールする前に、Web エージェントが URL のクエリ データをデコードするように設定するには、DecodeQueryData パラメータの値を yes に設定します。

URL の最大サイズの設定

以下のパラメータを使用して、Web エージェントが処理できる最大 URL サイズを増加することができます。

MaxUrlSize

Web エージェントが処理できる URL の最大サイズ（バイト単位）を指定します。 Web サーバによって、URL の長さ制限は異なるため、このパラメータを設定する前に Web サーバ ベンダーのマニュアルを確認してください。

デフォルト： 4096 B

最大 URL サイズを変更するには、MaxUrlSize パラメータに指定されたバイトの数を変更します。

IIS Web サーバの設定

以下のいずれかの設定を使用して、IIS 用エージェントを管理します。

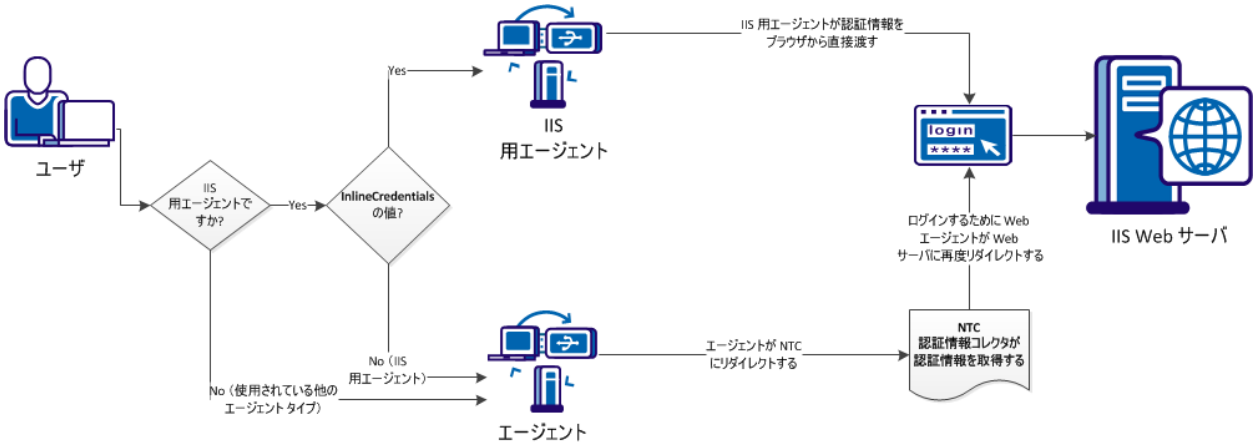
- InlineCredentials パラメータを使用して、統合 Windows 認証（IWA）リダイレクトを削除します (P. 345)。
- IIS サーバ ログにユーザ名およびトランザクション ID を記録します (P. 347)。
- IIS 認証に NetBIOS 名または UPN を使用します (P. 349)。
- NT チャレンジ/レスポンス認証を設定します (P. 350)。
- Information Card 認証方式を実装します (P. 357)。
- Information Card 認証方式の FCC テンプレートを設定します (P. 359)。
- IIS 7.x モジュール実行順序を制御します (P. 361)。
- IIS プロキシ ユーザ アカウントを使用します (P. 363)。
- 匿名ユーザ アクセスを有効にします (P. 364)。
- IIS 用エージェント上の Windows セキュリティ コンテキストを無効にします (P. 365)。
-

NTLM 認証情報コレクタ（NTC）にリダイレクトせずに、ユーザ認証情報を取得するように IIS 用のエージェントを設定

デフォルトでは、SiteMinder エージェントは、Windows 認証方式により保護されたリソースへの要求を、Windows 認証情報を取得するために NTLM 認証情報コレクタ（NTC）へリダイレクトします。

ユーザの認証情報を HTTP 要求からインラインで（すなわち、NTC にリダイレクトしないで）取得するように、IIS の SiteMinder エージェントを設定できます。

以下の図は、2 つの認証情報収集方法間の違いについて説明しています。



NTC にリダイレクトせずに、HTTP 要求からユーザの認証情報を取得するようにエージェントを設定するには、以下のように InlineCredentials 設定パラメータを設定します。

InlineCredentials

IIS 用エージェントがどのようにユーザ認証情報を処理するかを指定します。 このパラメータの値が yes の場合、IIS 用エージェントは HTTP 要求から認証情報を直接読み取ります。このパラメータの値が No の場合、エージェントは NTC 認証情報コレクタにリダイレクトします。

デフォルト： No

注: 環境内のいずれかの SiteMinder エージェントが NTC リダイレクトを使用するように設定されている場合は、NT チャレンジ/レスポンス認証を設定します。

詳細情報:

IIS が NT チャレンジ/レスポンス認証をサポートするようにエージェントを設定する (P. 350)

IIS サーバ ログでのユーザ名およびトランザクション ID の記録

Web エージェントは、ユーザ許可リクエストが成功するたびに、一意のトランザクション ID を生成します。 エージェントは、HTTP ヘッダにその ID を追加します。 ID は以下のログにも記録されます。

- 監査ログ
- Web サーバ ログ（サーバがクエリ文字列をログに記録するように設定されている場合）
- ポリシー サーバ ログ

トランザクション ID を使用して、所定のアプリケーションのユーザ アクティビティを追跡できます。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

トランザクション ID は、モック クエリ パラメータとしてログに表示され、既存のクエリ文字列の末尾に追加されます。 以下の例に、クエリ文字列（末尾は STATE=MA）に追加されたトランザクション ID（太字）を示します。

```
172.24.12.1, user1, 2/11/0, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realm/index.html,
STATE=MA&SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1
```

URL にクエリ パラメータがない場合、エージェントはトランザクション ID を Web サーバ ログ エントリの末尾に追加します。 以下に例を示します。

```
172.24.12.1, user1, 2/11/0, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realma/index.html,
SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1.
```

注: ユーザがリソースにアクセスすると、Web エージェントは、ユーザ名とアクセス情報をネイティブの Web サーバ ログ ファイルに記録します。

IIS サーバ上のリソースを保護するエージェントは、デフォルトでは SiteMinder トランザクション ID や認証されたユーザ名を IIS サーバ ログに記録 *しません*。 これらのエージェントは ISAPI 拡張（クラシック パイプライン モード）として動作できるので、サーバがすでにトランザクションを記録しています。 以下のパラメータを使用して、この情報を追加するようにエージェントに強制することができます。

AppendIISServerLog

認証されたユーザ名と SiteMinder トランザクション ID を、IIS サーバ ログの cs-uri- query フィールドに追加するように、エージェントに指示します。 クエリ文字列が含まれる URL については、IIS サーバ ログの cs-uri- query フィールドのクエリ文字列、SiteMinder トランザクション ID およびユーザ名がカンマで区切られます。

デフォルト： No

SetRemoteUser

レガシー アプリケーションで必要となる可能性がある *REMOTE_USER HTTP* ヘッダ変数の値を指定します。

デフォルト： No

IIS サーバ ログにトランザクション ID およびユーザ名を記録するには、以下の手順に従います。

1. AppendIISServerLog パラメータの値を yes に設定します。
2. SetRemoteUser パラメータの値を yes に設定します。

   ユーザ名およびトランザクション ID が IIS サーバ ログに表示されます。

IIS 認証での NetBIOS 名または UPN の使用

IIS ネットワークでは、要求されたリソースの場所に関して、ドメイン名とは異なる NetBIOS 名が存在する場合があります。保護されたリソースにアクセスを試みたときに複数のドメイン コントローラが存在すると、ユーザ認証は失敗し、Web サーバ ログに「IIS ログオン エラー」と表示されます。以下のパラメータを使用して、UPN または NetBIOS 名を IIS Web サーバに送信するかどうかを制御できます。

UseNetBIOSforIISAuth

IIS 6.0 Web エージェントが IIS ユーザ認証のために、ユーザ プリンシパル名（UPN）と NetBIOS 名のどちらを IIS 6.0 Web サーバに送信するかを指定します。

注: このパラメータは、Active Directory ユーザ ストアがポリシー サーバに関連付けられている場合のみ有効です。

このパラメータを有効にした場合は、SiteMinder の認証時にポリシー サーバが Active Directory からユーザ DN、UPN、および NetBIOS 名を抽出し、このデータを IIS 6.0 Web エージェントに送り返します。

管理 UI でユーザ ディレクトリに対して［認証済みユーザ セキュリティ コンテキストで実行］オプションを選択したかどうか、および UseNetBIOSforIIAuth パラメータをどのように設定したかに応じて、ユーザのログオン認証情報は以下のように送信されます。

- UseNetBIOSforIISAuth パラメータが no に設定されている場合、IIS 6.0 Web エージェントは UPN 名を送信します。
- UseNetBIOSforIISAuth パラメータが yes に設定されている場合、Web エージェントは NetBIOS 名を送信します。

IIS Web サーバは、Web エージェントから受け取った認証情報を使用してユーザを認証します。

デフォルト： No

Web エージェントで IIS 認証の NetBIOS 名が使用されるようにするには、UseNetBIOSAuth パラメータを yes に設定します。

IIS が NT チャレンジ/レスポンス認証をサポートするようにエージェントを設定する

環境内のいずれかの SiteMinder エージェントが NTC リダイレクトを使用するように設定されている場合は、NT チャレンジ/レスポンス認証を設定します。

ユーザがリソースへのアクセスを要求するときに、IIS Web サーバは NT チャレンジ/レスポンス認証を使用して、そのユーザの Internet Explorer ブラウザにチャレンジします。

注: NT チャレンジ/レスポンス認証は、Internet Explorer ブラウザのみと連携します。

以下の方法のどちらかで NT チャレンジ/レスポンス認証を実装できます。

- 保護されているリソースにユーザがアクセスしようとしたときに、そのユーザにチャレンジする。シングル サインオン環境のユーザに対しては、そのユーザが初めてリソースを要求したときだけチャレンジする。
- ユーザに使用している Internet Explorer ブラウザの自動ログオン機能を設定させる。

  自動ログオン機能を使用すると、ユーザはチャレンジを受けずにリソースにアクセスできるようになります。 認証処理は引き続き実行されますが、ブラウザとサーバの間の NT チャレンジ/レスポンス処理はユーザには見えません。 一般的に、自動ログオンはイントラネットで使用されます。イントラネットでは、セキュリティがそれほど厳重ではないので、ユーザがリソースにシームレスにアクセスできるようにします。 自動ログオンは、インターネットを介した通信にはお勧めできません。

SiteMinder エージェントは認証情報コレクタを使用して、NT チャレンジ/レスポンス認証方式のために、ユーザの Windows 認証情報を収集します。エージェントでは、NTLM 認証情報を収集する目的で、NTC 拡張子をサポートしています。

注: このデフォルト動作を変更する場合にのみ、NTCEXT を設定します。

SiteMinder に NT チャレンジ/レスポンス認証を使って処理させるには、以下の手順に従います。

1. 次のタスクによって、IIS Web サーバの NT チャレンジ/レスポンス認証を設定します。
   a. ファイル拡張子 .ntc をマップします (P. 351)。
   b. 仮想ディレクトリを作成および設定し、その仮想ディレクトリが NT チャレンジおよびレスポンス 認証情報を求めることを確認します (P. 352)。
2. 管理 UI で、NT チャレンジ/レスポンス認証の Windows 認証方式を設定します (P. 354)。
3. NTLM 認証情報コレクタを指定します (P. 229)。
4. 管理 UI を使用して、NT チャレンジ/レスポンス認証に関するポリシーを設定します。

   注: 詳細については、「ポリシー サーバ設定ガイド」を参照してください。
5. （オプション）ユーザが使用している Internet Explorer ブラウザの自動ログオン機能を設定させます (P. 355)。

   IIS の NT チャレンジ/レスポンス認証が設定されます。

詳細情報

NTLM 認証情報コレクタ（NTC）にリダイレクトせずに、ユーザ認証情報を取得するように IIS 用のエージェントを設定 (P. 345)

ファイル拡張子 .NTC のマップ

IIS Web サーバ上で NT チャレンジ/レスポンス認証を設定するには、ISAPIWebAgent.dll アプリケーションにファイル拡張子 .NTC をマップします。

ファイル拡張子 .NTC をマップする方法

1. インターネット サービス マネージャを開きます。
2. 左ペインの［Web サイト］を右クリックし、右ペインの［既定の Web サイト］を右クリックして［プロパティ］を選択します。

   ［既定の Web サイトのプロパティ］ダイアログ ボックスが表示されます。
3. ［ホーム ディレクトリ］タブをクリックします。
4. ［アプリケーションの設定］セクションで［構成］をクリックします。

   ［アプリケーションの構成］ダイアログ ボックスが表示されます。
5. ［追加］をクリックします。

   ［アプリケーションの拡張子マッピングの追加/編集］ダイアログ ボックスが開きます。

   a. ［実行可能ファイル］フィールドで、［参照］をクリックし、次のファイルを見つけます。*web_agent_home*/bin/ISAPIWebAgent.dll。

   b. ［開く］をクリックします。

   c. ［拡張子］フィールドに「.ntc」と入力します。
6. ［OK］を 3 回クリックします。

   ［アプリケーションの拡張子マッピングの追加/編集］ダイアログ ボックス、［アプリケーション構成］ダイアログ ボックス、および［既定の Web サイトのプロパティ］ダイアログ ボックスが閉じます。ファイル拡張子 .ntc がマップされます。

Windows 認証方式用の仮想ディレクトリの作成と設定（IIS 7.5）

SiteMinderWindows 認証方式を使用するには、IIS 7.x Web サーバ上で仮想ディレクトリを設定します。 仮想ディレクトリでは、認証情報に関して Windows チャレンジおよびレスポンスが必要です。

次の手順に従ってください:

1. インターネット インフォメーション サービス（IIS）マネージャを開きます。
2. 左側のペインで、以下のアイテムを展開します。
   - Web サーバ アイコン
   - サイト フォルダ
   - 既定の Web サイト アイコン
3. siteminderagent 仮想ディレクトリを右クリックし、次に、［仮想ディレクトリの追加］を選択します。

   ［仮想ディレクトリの追加］ダイアログ ボックスが表示されます。
4. ［エイリアス］フィールドに、以下の値を入力します。

   `ntlm`
5. ［参照］ボタン（［物理パス］フィールドの横）をクリックし、次に、以下のディレクトリを見つけます。

   web_agent_home¥samples

   仮想ディレクトリが作成されます。
6. 以下のいずれかの手順で仮想ディレクトリを設定します。
   - SiteMinder Windows 認証方式で Web サイト全体のリソースをすべて保護するには、［既定の Web サイト］アイコンをクリックします。
   - SiteMinder Windows 認証方式で Web サイト全体を保護しない場合は、ntlm 仮想ディレクトリ（手順 4 で作成済み）をクリックします。
7. ［認証］アイコンをダブル クリックします。

   ［認証］ダイアログ ボックスが表示されます。
8. 以下の手順を実行します。
   a. ［匿名認証］を右クリックし、次に、［無効］を選択します。
   b. ［Windows 認証］を右クリックし、次に、［有効］を選択します。

      Windows 認証方式用の仮想ディレクトリが設定されます。

注: Web サーバを再起動すると、これらの変更が反映されます。

チャレンジ/レスポンス認証の Windows 認証方式の設定

NT チャレンジ/レスポンス認証を実装するには、以下の値を持った Windows 認証方式を設定するポリシー管理者を提供します。

Server Name

IIS Web サーバの完全修飾ドメイン名。たとえば次のようになります。

server1.myorg.com

ターゲット

/siteminderagent/ntlm/smntlm.ntc

注: このディレクトリは、インストール時にすでに設定された仮想ディレクトリと一致している必要があります。 ターゲットである smntlm.ntc は、存在していなくてもかまいません。また、.ntc で終わる名前や、デフォルトの代わりに使用するカスタム MIME タイプでもかまいません。

ライブラリ

smauthntlm

詳細情報

認証情報コレクタの MIME タイプ (P. 196)

NTLM 認証情報コレクタの指定

NTLM 認証情報コレクタ（NTC）は、Web エージェント内のアプリケーションです。 NTC は、Windows 認証方式によって保護されるリソースに関連する NT 認証情報を収集します。 この方式は、Internet Explorer ブラウザからアクセスされる、IIS Web サーバ上のリソースに適用できます。

各認証情報コレクタには、1 つの MIME タイプが関連付けられています。IIS に関しては、以下のパラメータで NTC MIME TYPE が定義されています。

NTCExt

NTLM 認証情報コレクタと関連付けられた MIME タイプを指定します。 このコレクタは、Windows 認証方式によって保護されているリソースに関連する NT 認証情報を収集します。 この方式は、Internet Explorer ブラウザのユーザのみがアクセスする IIS Web サーバ上のリソースに適用できます。

このパラメータに複数の拡張子を持たせることもできます。 エージェント設定オブジェクトを使用している場合は、複数値オプションを選択します。 ローカル設定ファイルを使用している場合は、各拡張子をカンマで区切ります。

デフォルト： .ntc

使用している環境で NTCExt パラメータによって指定されたデフォルトの拡張子がすでに使用されている場合は、別の MIME タイプを指定できます。

認証情報コレクタをトリガする拡張子を変更するには、他のファイル拡張子を NTCExt パラメータに追加します。

Internet Explorer に対する自動ログオンの設定

エージェントがユーザにクレデンシャルをチャレンジすることなくユーザを認証するには、Internet Explorer ブラウザのユーザは自動ログオン ブラウザ セキュリティ設定を設定する必要があります。

次の手順に従ってください:

1. Internet Explorer ブラウザを起動します。
2. ［インターネット オプション］ダイアログ ボックスを開きます（使用しているブラウザのバージョンでダイアログ ボックスを開く方法については、Internet Explorer のオンライン ヘルプを参照してください）。
3. ［セキュリティ］タブをクリックします。
4. 正しいセキュリティ ゾーンをクリックします。

5. カスタム レベルをクリックします。
6. ［ユーザ認証］セクションまでスクロール ダウンします。［ログオン］オプションで、4.x の場合は［現在のユーザ名とパスワードで自動的にログオン］オプションを、5.x または 6.0 の場合は［現在のユーザ名とパスワードで自動的にログオンする］オプションをオンにします。
7. 変更を適用します。

   ［セキュリティ設定］ダイアログ ボックスおよび［インターネット オプション］ダイアログ ボックスが閉じます。ユーザの設定が保存され、自動ログインが設定されます。

Information Card 認証方式を実装する方法

CA SiteMinder では、Windows CardSpace を実装する Information Card 認証方式（ICAS） をサポートします。 保護されているリソースへのアクセスを求めるユーザは認証カードを選択できます。 SiteMinder では、ユーザの身元を確認するためにカードに含まれている情報が使用されます。

ICAS の実装では、以下の SiteMinder コンポーネントの設定変更が必要です。

- SiteMinder Web エージェントをホストしているサーバ
- SiteMinder ポリシー サーバ
- smkey データベース

次の手順に従ってください:

1. Web サーバ上で以下のタスクを実行します。
   a. IIS Web サーバ上で SSL 通信を有効にします。

      注: 詳細については、Microsoft のドキュメントを参照するか、または http://support.microsoft.com/ に移動します。

   b. Web サーバ証明書を .pfx ファイルとしてエクスポートします。
   c. SiteMinder InfoCard.fcc テンプレートをカスタマイズします。
2. ポリシー サーバ上で以下のタスクを実行します。
   a. ポリシー サーバに JCE をインストールします。
   b. ポリシー サーバ上の java.security ファイルを更新します。
   c. ポリシー サーバ上の config.properties ファイルを更新します。
   d. smkey データベースがまだない場合は、ポリシー サーバ設定ウィザードで作成します。
   e. Web サーバから smkey データベースに.pfx ファイル証明書を追加します。
   f. ポリシー サーバ内のユーザ ディレクトリを設定します。
   g. 管理 UI を使用して CardSpace のカスタム認証方式を作成します。
   h. （オプション）レスポンスで使用するセッション ストア内に、要求を格納します。
   i. （オプション）セッション ストアから要求値を取得できるようにすることにより、パーソナライズを有効にします。

j. （オプション）格納された要求値を取得するようにアクティブなレスポンスを設定します。

Information Card 認証方式のための FCC のテンプレートの設定

SiteMinder Web エージェントには、SiteMinder 内の ICAS を実装するために使用できるフォーム認証情報コレクタ（FCC）のテンプレートが含まれます。

次の手順に従ってください:

1. テキスト エディタで以下のデフォルトの FCC ファイルを開きます。

   web_agent_home¥samples_default¥forms¥InfoCard.fcc

2. 以下のディレクトリにファイルのコピーを保存します（コピーを作成することにより、後で必要になった場合に備えて、デフォルトの FCC 設定が維持されます）。

   web_agent_home¥samples¥forms¥

3. FCC のファイルのコピーから、以下の情報を記録します。

   重要: この情報は、ポリシー サーバを設定するために必要です。

   - Web エージェントをホストしている IIS Web サーバの完全修飾ドメイン名
   - 手順 2 で保存した FCC ファイルの名前
   - 手順 2 で保存した FCC ファイル内の requiredClaims パラメータ タグの値（引用符なし） 以下の例を参照してください。

     ```
     http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier
     ```

   - （オプション）ホワイト リスト処理が実行されたときの保証レベル（LOA）の requiredClaims パラメータ タグの値 以下の例を参照してください。

     ```
     < param name="requiredClaims" value="
     http://idmanagement.gov/icam/2009/09/imi_1.0_profile#assurancelevel1 ”/>
     ```

     さまざまな LOA の URI は次のとおりです。

     ```
     http://idmanagement.gov/icam/2009/09/imi_1.0_profile#assurancelevel1
     http://idmanagement.gov/icam/2009/09/imi_1.0_profile#assurancelevel2
     http://idmanagement.gov/icam/2009/09/imi_1.0_profile#assurancelevel3
     ```

4. （オプション） テキスト エディタを使用して、FCC ファイルのコピーで以下の変更のいずれかを加えます。

   - カスタム ロゴを使用するには、netegrity_logo.gif ファイルを独自の画像に置換し、FCC ファイル内の以下のリンクをそれに応じて更新します。

```
<img alt="Logo" src="/siteminderagent/dmspages/netegrity_logo.gif">
```

IIS 用 SiteMinder エージェント使用時の IIS 7.x モジュール実行順序の制御

IIS Web サーバに IIS 用 SiteMinder エージェント をインストールし設定するとき、IIS 用エージェントは他のモジュールの前に実行されます。 IIS 環境で別のモジュールを最初に実行することが必要な場合、Windows レジストリの以下の場所に設定された番号を変更できます。

```
HKLM¥SOFTWARE¥Wow6432Node¥Netegrity¥SiteMinder Web Agent¥Microsoft IIS¥RequestPriority
```

たとえば、IIS 7.x Web サーバ内の他のモジュール（UrlScan など）に、IIS 用 SiteMinder エージェント と同じ実行優先度が割り当てられているとします。この設定を使用して SiteMinder モジュールがいつ実行されるかを制御します。

次の手順に従ってください:

1. IIS Web サーバ上で Windows レジストリ エディタを開きます。
2. 以下のキーを展開します。

   ```
   HKLM¥SOFTWARE¥Wow6432Node¥Netegrity¥SiteMinder Web Agent¥Microsoft IIS
   ```

3. 以下の値を見つけます。

   ```
   RequestPriority
   ```

4. RequestPriority の値を、次の値に対応する望みの数値に変更します。

   PRIORITY_ALIAS_FIRST

   IIS 用 SiteMinder エージェントを、IIS Web サーバの他のモジュールの前に実行します。 これがデフォルトの設定です。

   例： 0 （最初）

   デフォルト： 0

   PRIORITY_ALIAS_HIGH

   最初に実行すると設定された任意のモジュールの後で、かつ、実行優先度が「中」、「低」または「最後」 に設定されたモジュールよりも先に、IIS 用 SiteMinder エージェントのモジュールを実行します。

   例： 1 （高）

   PRIORITY_ALIAS_MEDIUM

   IIS 用 SiteMinder エージェントの各モジュールは、first および high での実行を設定されたモジュールの後で、かつ、low または last 優先度の実行が設定されたモジュールの前に実行します。

例： 2 （Medium）

PRIORITY_ALIAS_LOW

IIS 用 SiteMinder エージェントの各モジュールは、first、high、および medium での実行を設定されたモジュールの後で、かつ、last 優先度の実行が設定されたモジュールの前に実行します。

例： 3 （Low）

PRIORITY_ALIAS_LAST

他のすべてのモジュールの*後に* IIS 用 SiteMinder エージェントのモジュールを実行します。

例： 4 （Last）

5. 変更を保存し、レジストリ エディタを閉じます。
6. 設定をテストし、IIS モジュール用エージェントが実行される前に望むモジュールが実行されることを確認します。

IIS プロキシ ユーザ アカウントの使用（IIS のみ）

SiteMinder によって保護された IIS Web サーバ上のリソースにアクセスしようとしたユーザにそれらのリソースに対する十分な IIS 権限がない場合、Web エージェントがアクセスを拒否することがあります。たとえば、UNIX システム上の LDAP ユーザ ディレクトリに格納されているユーザは、IIS Web サーバを持つ Windows システムにアクセスできないことがあります。

SiteMinder からアクセス権を付与されたユーザは、IIS Web サーバによって十分な権限のあるデフォルトのプロキシ アカウントが得られます。 ユーザが有効な Windows セキュリティ コンテキストを持つ場合でも、Web エージェントは DefaultUserName および DefaultPassword パラメータの値を認証情報として使用します。

次の手順に従ってください:

1. ForceIISProxyUser パラメータの値を以下の値の*いずれか*に設定します。
   - IIS サーバ上のアプリケーションへのアクセスがユーザの認証情報自体に基づく場合は、ForceIISProxyUser パラメータの値を yes に設定します。
   - IIS サーバ上のアプリケーションへのアクセスがユーザの代わりに動作する特定のアカウント（プロキシなど）に基づく場合は、ForceIISProxyUser パラメータの値を no に設定します。

   デフォルト： No
2. 以下の Windows 機能のどちらも*使用していない*場合は、手順 3 に進みます。
   - Windows 認証方式
   - Windows ユーザ セキュリティ コンテキスト
3. DefaultUserName パラメータ内にプロキシ ユーザ アカウントのユーザ名を入力します。 ドメイン アカウントと、そのドメインの一部ではないローカル マシンを使用している場合は、以下の例に示す構文を使用します。

   DefaultUserName=*Windows_domain¥acct_with_admin_privilege*

   それ以外の場合は、ユーザ名のみを指定します。
4. DefaultPassword パラメータ内に既存の Windows ユーザ アカウントに関連付けられたパスワードを入力します。

重要: 暗号化できるので、エージェント設定オブジェクト内でこのパラメータを設定することをお勧めします。ローカル設定ファイル内で設定すると、値は暗号化されずにプレーン テキストで格納されます。

IIS Proxy アカウントが設定されます。

匿名ユーザ アクセスの有効化

ユーザにプロキシ ユーザとしてのアクセス権を付与しない場合は、以下のパラメータを設定します。

UseAnonAccess

プロキシ ユーザの認証情報を使用するのではなく、匿名ユーザとして Web アプリケーションを実行するように IIS Web エージェントに指示します。

デフォルト： No

注: このパラメータは IIS Web エージェントにのみ適用されます。

匿名ユーザ アクセスを有効にするには、UseAnonAccess パラメータを yes に設定します。

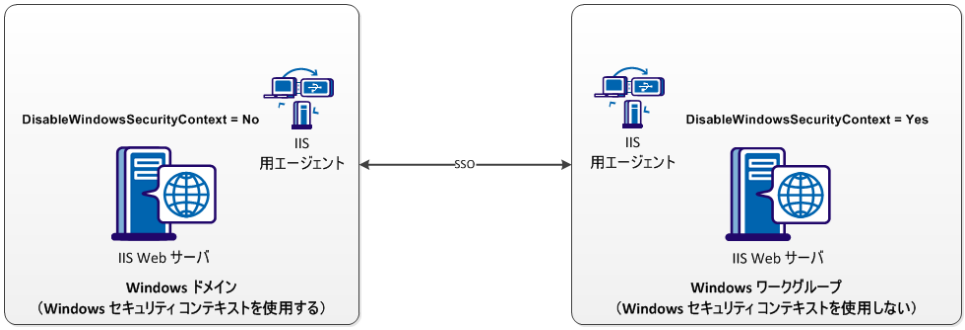
IIS 用エージェント上の Windows セキュリティ コンテキストの無効化

SiteMinder ポリシー サーバは、ユーザのセッションから Windows セキュリティ コンテキストを取得します。 ほとんどの状況では、セッション情報がすべてのエージェントに利用可能なため、この環境はシングル サインオンに受け入れられます。

以下に、シングル サインオンに別の設定が必要な状況例を示します。

- ある SiteMinder エージェントは Windows セキュリティ コンテキストを使用します。
- 別の SiteMinder エージェントは Windows セキュリティ コンテキストを使用しません。

この状況を次の図に示します。



Windows セキュリティ コンテキストを使用する、Windows ドメインと、Windows セキュリティ コンテキストを使用しない Windows ワークグループの間の SSO を許可するには、以下のパラメータを設定します

DisableWindowsSecurityContext

エージェントの Windows セキュリティ コンテキストを無効にします。 このパラメータの値が yes の場合、エージェントはユーザの Windows セキュリティ コンテキストを無視します。 このパラメータの値が false または no の場合、エージェントはユーザのセッションに含まれている Windows セキュリティ コンテキストを使用します。 このパラメータは、セキュリティ コンテキストを使用する Windows 環境と使用しない Windows 環境間のシングル サインオンを可能にします。

デフォルト： False

制限： Yes, No

Cookie が含まれるサーバ レスポンスのキャッシュの防止

IIS Web サーバは出力キャッシュを使用して、それらのレスポンスを格納します。 エージェントへのレスポンスには Cookie が含まれます。 IIS Web サーバがその出力キャッシュから認証レスポンスを送信する場合、別のユーザがキャッシュされたレスポンスで認証 Cookie を受信する可能性があります。

たとえば、ユーザ 1 は正常に認証を行い、IIS サーバは Cookie を持ったレスポンスをキャッシュします。ユーザ 2 がユーザ 1 と同じリソースにアクセスする場合、IIS Web サーバは恐らくユーザに対して応答を返す可能性があります‥‥ 1 〜ユーザ 2。

製品は、デフォルトで Cookie が含まれているアイテムの IIS 出力キャッシュを無効にします。 後方互換性の為に、製品の旧バージョンの動作に戻す場合は、以下のパラメータの値を「no」に変更します。

IISCacheDisable

IIS Web サーバが、出力キャッシュに Cookie が含まれるレスポンスを格納するかどうかを指定します。 SiteMinder 処理が発生する前に、IIS Web サーバはキャッシュされた応答を送信します。 出力キャッシュを無効にすると、IIS による各トランザクションの認証および許可が強制的に実行されます。 パラメータの値を［はい］に設定すると、あるユーザが別のユーザ宛の認証および許可レスポンスを誤って受け取るのを阻止できます。

デフォルト： はい（キャッシュ無効）

IIS 用のエージェントの Cookie 設定タイミングの決定

IIS 用の SiteMinder エージェントは、IIS 7.x Web サーバが提供する Application Request Routing（ARR）機能をサポートします。ARR は、他の Web サーバ ベンダーによって提供されるリバース プロキシ サーバ機能に類似した Microsoft IIS Web サーバで作動します。

SiteMinder エージェントはすべて Cookie を処理します。Cookie 処理をいつ行うか制御する状況には、以下のような条件があります。

- ARR が使用される。
- FCCCompatMode エージェント設定パラメータの値が yes に設定されている。
- カスタム エージェント（SiteMinder SDK で開発されていた）を使用しています。

エージェントが Cookie を処理するタイミングの制御は、SiteMinder 保護レベルを適用することでセキュリティを維持します。

SiteMinder エージェントの一部の展開では、トランザクションの特定のポイントでの SiteMinder Cookie 処理が必要です。SiteMinder エージェントはすべて、Cookie を使用し処理します。 状況によっては、トランザクションのより早い段階で Cookie の処理が必要になります。 より遅い段階で Cookie の処理が必要な状況もあります。 適切な時間に Cookie を処理することで、SiteMinder がユーザのリソースを適切に保護していることが確認されます。

重要: 間違ったタイミングで Cookie を処理すると、保護レベルに影響します。 ARR 機能が実行する追加処理には、SiteMinder エージェントが Cookie を処理する相対時間を変更する必要があります。

次の手順に従ってください:

1. 管理 UI から、目的のエージェント設定オブジェクトを開きます。
2. 以下のパラメータを探します。

   EarlyCookieCommit

   Cookie が処理の初期段階で設定されるか、それとも後の段階で設定されるかを指定します。以下の条件のいずれかが存在する場合、このパラメータの値を［yes］に設定します。

   - IIS Web サーバは、アプリケーション リクエスト ルーティング（ARR）を使用します。
   - FCCCompatMode パラメータの値を［yes］に設定します。
   - カスタム エージェント（SiteMinder SDK で開発されていた）を使用しています。

   この値が［yes］である場合、OnAuthenticateRequest または OnPostAuthenticateRequest 通知方法を使用して、Web エージェントがリクエストを処理した後、Cookie はコミットされます（早めに）。

   早めの Cookie 処理を必要とする任意のカスタム アプリケーションに対して以前の SiteMinder エージェントの動作を保存しておくには、このパラメータの値を［yes］に設定します。

   この値が［no］である場合、Cookie は、OnSendResponse リクエスト通知メソッド中に、パイプラインの最後でレスポンス へ（後で）コミットされます。

   制限： IIS 7.x 専用エージェント。 この設定は、統合パイプライン モードを使用する Web アプリケーションのみをサポートします。

   デフォルト： No（Cookie は、OnSendResponse リクエスト通知メソッド中に、（後で）設定されます）。
3. 値フィールドをクリックしてから、前のパラメータの値を「yes」に変更します。
4. ［OK］をクリックします。
5. ［サブミット］をクリックします。

   確認メッセージが表示されます。

Apache Web サーバの設定

Apache ベースのサーバ用の SiteMinder エージェントを管理するために、以下の設定項目のうち、必要なものを設定します。

- HttpsPorts パラメータを設定します (P. 369)。
- レガシー アプリケーションを使用します (P. 370)。
- ポート番号に対して HTTPHostRequest パラメータを使用します (P. 370)。
- Apache Web サーバ ログにトランザクション ID を記録します (P. 371)。
- コンテンツ タイプが POST 要求でどのように転送されるか選択します (P. 372)。
- IPC セマフォ関連メッセージの Apache エラー ログへの出力を制限します (P. 373)。
- Stronghold サーバから証明書を削除します (P. 374)。

Apache 2.x サーバ上での HttpsPorts パラメータの使用

Apache 2.x Web サーバで、SSL アクセラレータ、または HTTP_HOST ヘッダの値を変更するいずれかの中間デバイスを使用しており、HttpsPorts パラメータを使用する場合は、追加の Web サーバ設定変更が必要です。

次の手順に従ってください:

1. Apache Web サーバの httpd.conf ファイルを開き、以下の変更を加えます。
   - UseCanonicalName パラメータの値を on に変更します。
   - ServerName パラメータの値を以下のように変更します。

     server_name:*port_number*

     server_name

     SSL アクセラレータのホスト名を指定します。
2. Web エージェントの以下の設定パラメータを変更します。
   - GetPortFromHeaders パラメータの値を yes に変更します。

Apache Web エージェントでのレガシー アプリケーションの使用

（HTTP 1.1 をサポートしない）レガシー アプリケーションがあり、それらを Apache Web サーバで実行する場合は、以下のパラメータを設定します。

LegacyTransferEncodingBehavior

Web エージェントが使用するメッセージ エンコーディングのタイプを指定します。 このパラメータの値が no の場合、転送エンコーディング（transfer-encoding）がサポートされます。

このパラメータの値が yes の場合、コンテンツ エンコーディングがサポートされます。 transfer-encoding ヘッダは無視され、content-length ヘッダのみがサポートされます。

デフォルト： No

Apache Web サーバでレガシー アプリケーションを使用するには、LegacyTransferEncodingBehavior パラメータの値を yes に設定します。

重要: このパラメータの値を yes に設定すると、Federation や、4 KB より長い POST データの維持といった機能が動作せず、大きな証明書が認識されない場合があります。

ポート番号に関する HTTP HOST 要求の使用

実際の HTTP ヘッダを*変更せずに*、特定の Web サーバへのトラフィックをリダイレクトすることにより、負荷分散を実行するアプリケーションがある場合は、以下のパラメータを使用して、（ロード バランサによって使用されるポートの代わりに）適切な外部ポートにユーザをリダイレクトするように Web エージェントを設定する必要があります。

GetPortFromHeaders

Web サーバ サービス構造からポート番号を取得する代わりに、HTTP HOST リクエスト ヘッダからポート番号を取得するように Web エージェントに指示します。

デフォルト： No

注: このパラメータは、Apache Web エージェントにとって必須です。

HTTP HOST 要求ヘッダ内でポート番号を使用するには、GetPortFromHeaders パラメータを yes に設定します。

Apache Web サーバ ログへのトランザクション ID の記録

Web エージェントは、ユーザ許可リクエストが成功するたびに、一意のトランザクション ID を生成します。 エージェントは、HTTP ヘッダにその ID を追加します。 ID は以下のログにも記録されます。

- 監査ログ
- Web サーバ ログ（サーバがクエリ文字列をログに記録するように設定されている場合）
- ポリシー サーバ ログ

トランザクション ID を使用して、所定のアプリケーションのユーザ アクティビティを追跡できます。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

トランザクション ID は、モック クエリ パラメータとしてログに表示され、既存のクエリ文字列の末尾に追加されます。 以下の例に、クエリ文字列（末尾は STATE=MA）に追加されたトランザクション ID（太字）を示します。

```
172.24.12.1, user1, 2/11/0, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realm/index.html,
STATE=MA&SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1
```

URL にクエリ パラメータがない場合、エージェントはトランザクション ID を Web サーバ ログ エントリの末尾に追加します。 以下に例を示します。

```
172.24.12.1, user1, 2/11/0, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realma/index.html,
SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1.
```

注: ユーザがリソースにアクセスすると、Web エージェントは、ユーザ名とアクセス情報をネイティブの Web サーバ ログ ファイルに記録します。

Apache Web サーバ ログの SMTRANSACTIONID ヘッダ変数に SiteMinder トランザクション ID を記録できます。

次の手順に従ってください:

1. httpd.conf ファイルを開きます。
2. LogFormat ディレクティブに SM_TRANSACTIONID ヘッダ変数を追加します。

   例：

   ```
   LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{SM_TRANSACTIONID}i¥"" common
   ```

   注: httpd.conf ファイルおよび LogFormat ディレクティブの詳細については、Apache Web サーバのマニュアルを参照してください。
3. 変更を適用するにはサーバを再起動します。

   トランザクション ID が Apache Web サーバ ログに記録されます。

POST 要求でのコンテンツ タイプの転送方法の選択

Apache Web サーバを使用している場合、POST リクエストの間にコンテンツがサーバに転送される方法を、以下のパラメータで制御できます。

LegacyStreamingBehavior

コンテンツが POST 要求中にサーバにどのように転送されるかを指定します。 このパラメータの値が yes に設定されると、以下を*除く*すべてのコンテンツ タイプはストリームになります。

- text/xml
- application/x-www-form-urlencoded

このパラメータの値が no に設定されると、すべてのコンテンツ タイプがスプールされます。

デフォルト： No

POST 要求でのほとんどのタイプのコンテンツをストリームするには、LegacyStreamingBehavior パラメータの値を［はい］に変更します。

IPC セマフォ関連メッセージ出力の Apache エラー ログへの制限

デフォルトでは、Apache Web エージェントは、設定された Apache のロギング レベルにかかわらず、Apache のエラー ログへのすべてのレベル（情報およびエラー）の IPC セマフォ関連メッセージを記録します。

Web エージェントの IPC セマフォ関連出力の詳細を Apache のエラー ログに制限するには、*web_agent_home*/config 内にある trace.conf ファイルに以下のパラメータを追加します。

nete.stderr.loglevel

Web エージェントが Apache のエラー ログに記録する IPC セマフォ関連メッセージのレベルを指定します。 以下の値を受け入れます。

off

Web エージェントは、IPC セマフォ関連メッセージを Apache のエラー ログに記録しません。

error

Web エージェントは、IPC セマフォ関連のエラー メッセージのみを Apache のエラー ログに記録します。

info

（デフォルト）Web エージェントは、IPC セマフォ関連のエラーおよび情報メッセージを Apache のエラー ログに記録します。

例: trace.conf 内の nete.stderr.loglevel パラメータの定義

trace.conf の以下の抜粋では、IPC セマフォ関連のエラーメッセージのみが Apache のエラー ログに記録されるように Web エージェントを制限するように nete.stderr.loglevel パラメータが設定されています。

```
# CA Web Agent IPC logging levels
# nete.stderr.loglevel=error
```

Stronghold からの証明書の削除（Apache エージェントのみ）

Stronghold Web サーバはクライアント証明書をローカルの一時ファイル内に書き込みます。Web エージェントはこのファイルを使って証明書に基づく認証を行います。 Stronghold サーバはこのファイルを使って、クライアント証明書の情報を認証時に利用できるようにします。 ユーザが Web サイトにアクセスするたびにこれらの証明書ファイルは大きくなって、サーバのディスク領域を消費します。 Web エージェントが証明書ファイルを使用し終わったら削除するように、エージェントを設定することができます。

証明書ファイルを削除するには、DeleteCerts パラメータを yes に設定します。

Oracle iPlanet Web サーバの設定

Oracle iPlanet サーバ用 SiteMinder エージェントを管理するために、以下の設定項目のうち、必要なものを設定します。

- ディレクトリ参照を制限します (P. 375)。
- 複数の AuthTrans 関数を処理します (P. 376)。
- Oracle iPlanet Web サーバ ログにトランザクション ID を記録します (P. 377)。

Oracle iPlanet Web サーバ上でのディレクトリ参照の制限

Oracle iPlanet Web サーバのディレクトリを参照しようとするユーザが SiteMinder によって認証要求されるようにするために、以下のパラメータを設定できます。

DisableDirectoryList

最初に認証情報を要求せずに、ユーザがディレクトリの内容を表示または参照することを Web エージェントが認めるかどうかを指定します。 これは、以下の条件がすべて当てはまる場合に発生します。

- レルムがルート リソース（/）を保護するように設定されている。
- ディレクトリのデフォルト Web ページ（index.html など）が名前変更または削除されている。

デフォルト： No

Oracle iPlanet サーバ上のディレクトリ参照を制限する方法

1. エージェント設定オブジェクトまたはローカル設定ファイルに DisableDirectoryList パラメータを追加します。
2. DisableDirectoryList パラメータの値を yes に設定します。

   ディレクトリ参照が制限されます。 SiteMinder がディレクトリを参照しようとするユーザの認証を要求します。

Oracle iPlanet Web サーバでの複数の AuthTrans 関数の処理

AuthTrans 関数は、Oracle iPlanet Web サーバを初期化するためのディレクティブです。 Oracle iPlanet Web サーバは、obj.conf ファイル内に指定された順番に従って複数の AuthTrans 関数を実行します。 Oracle iPlanet サーバは、REQ_PROCEED コマンドが返されるまで、AuthTrans 関数を次々に呼び出します。 いったん REQ_PROCEED コマンドが返されると、それ以降の AuthTrans 関数は実行されません。

デフォルトでは、SiteMinder が最初の AuthTrans 関数になり、REQ_PROCEED を返します。 他の AuthTrans 関数が実行されるようにするには、EnableOtherAuthTrans パラメータを追加して値を yes に設定する必要があります。

このパラメータのデフォルト値は no です。複数の AuthTrans 関数を有効にするには、EnableOtherAuthTrans パラメータを yes に設定します。

このパラメータを追加することにより、SiteMinder Web エージェントが他の関数と共存できるようになります。

ただし、obj.conf ファイル内で、SiteMinder エージェントの関数を、AuthTrans ディレクティブの最初のエントリにしてください。 そのエントリは、次のようになります。

```
AuthTrans fn="SiteMinderAgent"
```

Oracle iPlanet Web サーバ ログのトランザクション ID の記録

Solaris で該当

Web エージェントは、ユーザ許可リクエストが成功するたびに、一意のトランザクション ID を生成します。 エージェントは、HTTP ヘッダにその ID を追加します。 ID は以下のログにも記録されます。

- 監査ログ
- Web サーバ ログ（サーバがクエリ文字列をログに記録するように設定されている場合）
- ポリシー サーバ ログ

トランザクション ID を使用して、所定のアプリケーションのユーザ アクティビティを追跡できます。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

トランザクション ID は、モック クエリ パラメータとしてログに表示され、既存のクエリ文字列の末尾に追加されます。 以下の例に、クエリ文字列（末尾は STATE=MA）に追加されたトランザクション ID（太字）を示します。

```
172.24.12.1, user1, 2/11/0, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realm/index.html,
STATE=MA&SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1
```

URL にクエリ パラメータがない場合、エージェントはトランザクション ID を Web サーバ ログ エントリの末尾に追加します。 以下に例を示します。

```
172.24.12.1, user1, 2/11/0, 15:30:10, W3SVC, MYSERVER, 192.168.100.100, 26844,
47, 101, 400, 123, GET, /realma/index.html,
SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1.
```

注: ユーザがリソースにアクセスすると、Web エージェントは、ユーザ名とアクセス情報をネイティブの Web サーバ ログ ファイルに記録します。

Oracle iPlanet Web サーバ ログに SiteMinder トランザクション ID を記録できます。

次の手順に従ってください:

1. magnus.conf ファイルを開きます。
2. 以下のヘッダ変数を、Web サーバ初期化時にロギングする HTTP サーバ変数の既存リスト内に追加します。

   ```
   %Req->headers.SM_TRANSACTIONID%"
   ```

   注: エージェント設定オブジェクトまたはローカル設定ファイル内で LowerCaseHTTP パラメータの値を yes に設定しなかった場合は、ヘッダ変数を大文字で入力します。

   以下の例では、SMTRANSACTIONID ヘッダ変数を既存のエントリに最後に太字で示しています。 ただし、変数のリスト内のどの場所にも配置できます。

   ```
   Init fn="flex-init" access="D:/iPlanet/server4/https-orion/logs/access"
   format.access="%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%]
   ¥" %Req->srvhdrs.clf-status% %Req-srvhdrs.content-length% %Req->headers.-
   SM_TRANSACTIONID%"
   ```

3. 変更を適用するため Oracle iPlanet サーバを再起動します。

   トランザクション ID が Oracle iPlanet Web サーバ ログに表示されます。以下の例は、Web サーバ ログのエントリを示しています。ここでは、トランザクション ID を太字で示しています。

   ```
   11.22.33.44 - user1 [21/Nov/2003:16:12:24 -0500] "GET /Anon/index.html HTTP/1.0"
   200 748 3890b4b9-58f8-4a74df53-07f6-0002df88
   ```

詳細情報:

ヘッダでの小文字 HTTP の使用（Oracle iPlanet、Apache、Domino Web サーバ） (P. 175)

Domino Web サーバの設定

Domino のサーバは時に特別の SiteMinder エージェントのパラメータを必要とします。 他に指示がない限り、これらのパラメータは Domino サーバにのみ使用されます。 Domino のリソースを保護するために、以下のカテゴリのトピックを使用します。

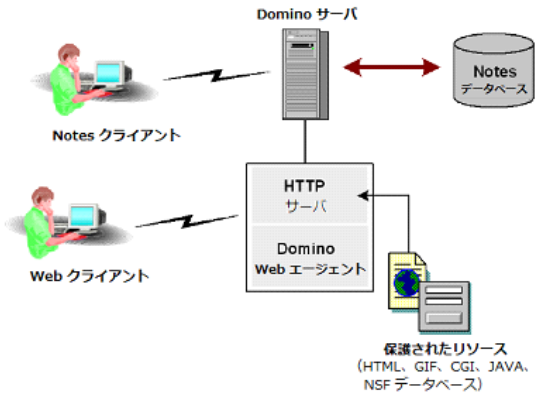
- 詳細については、次のトピックを参照してください。
  - Domino エージェントの概要 (P. 381)。
  - Domino URL 構文 x (P. 382)。
  - Domino のエイリアス (P. 383)。
- 基本的な設定情報については、次のトピックを参照してください。
  - Domino エージェントの設定 (P. 384)。
  - Domino 固有のエージェント機能の設定 (P. 385)。
  - Domino に関するユーザ ディレクトリの指定 (P. 385)。
  - Domino サーバに関するポリシーを作成する場合のガイドライン (P. 386)。
  - Domino のポリシーの設定 (P. 387)。
  - Domino サーバ リソースのルールの作成 (P. 388)。
- SiteMinder 認証の詳細については、次のトピックを参照してください。
  - Domino サーバによるユーザ認証 (P. 391)。
  - Domino スーパー ユーザとしての認証 (P. 392)。
  - 実ユーザまたはデフォルト ユーザとしての認証 (P. 393)。
  - Domino デフォルト ユーザおよび Domino スーパー ユーザの変更 (P. 394)。
  - Encryptkey ツールの使用による Domino デフォルト ユーザまたは Domino スーパー ユーザの設定 (P. 395)。
  - SiteMinder と Domino 認証の整合。
  - SiteMinder によるユーザ認証 (P. 396)。
  - SiteMinder ヘッダを使用した認証 (P. 397)。
  - Domino セッション認証の無効化 (P. 397)。
  - Domino での匿名 SiteMinder 認証方式の使用 (P. 398)。

- SiteMinder フォーム認証情報コレクタ (FCC) の使用の詳細については、次のトピックを参照してください。
  - 認証を目的とした Domino エージェントによる認証情報の収集 (P. 398)。
  - Domino エージェントによる FCC リダイレクト用 URL のマップ (P. 204)。
  - URL 正規化の無効化 (P. 400)。
- Lotus Notes ドキュメントへのアクセスの管理の詳細については、次のトピックを参照してください。
  - Lotus Notes ドキュメントへのアクセスの制御 (P. 402)。
  - Lotus Notes ドキュメント名の変換 (P. 403)。
- 前のリストで取り上げられていない件名の詳細については、次のトピックを参照してください。
  - Domino エージェントの完全ログオフ サポートの設定 (P. 404)。
  - Domino Web エージェントと WebSphere Application Server の連動。(P. 405)

Domino エージェントの概要

Domino アプリケーション サーバはメッセージング/Web アプリケーション プラットフォームであり、セキュリティ保護されたアクセスを LotusNotes クライアントに対して提供します。 Domino Web エージェントは、HTTP インタフェースである Domino アプリケーション サーバのみを保護し、HTML、JAVA、CGI などの Web リソースへのアクセス制御を行います。 Domino Web エージェントは Notes サーバを保護しません。

以下の図に、Domino Web エージェントが Domino サーバとどう統合されるかを示します。



Domino では、データは複数の Notes データベース内に保存されます。データベースに保存されるリソースとしては、ドキュメント、ビュー、フォーム、ナビゲータなど、さまざまな種類のオブジェクトが考えられます。これらのオブジェクトには、テキスト、ビデオ、グラフィック、オーディオなどのコンテンツを含めることができます。

Notes オブジェクトを開くには URL を使用します。 データベース内の Notes オブジェクトを Web 経由で利用できるように、Domino はオブジェクトから Web ページを動的に作成します。 データベース ビューの場合、Domino はビュー内の各ドキュメントへの URL リンクも作成します。Notes データベースからページを動的に作成することにより、最新の情報をユーザに提供できます。

Domino URL 構文

Domino サーバ上のリソースへのアクセスは URL に基づきます。 Domino サーバでは独自の URL 構文が使用されます。

Domino サーバは、次に例示するような標準的な URL を解釈できます。

```
http://www.example.com/index.html
```

Domino の URL コマンドは、以下の構文を使用できます。

```
http://host/database.nsf/Domino_object?Action_Argument
```

Host

サーバの DNS エントリまたは IP アドレスを示します。

Database

notes ¥data ディレクトリを基準とするパスまたはデータベース レプリカ ID で、データベース ファイル名を指定します。

Domino_object

ビュー、ドキュメント、フォーム、ナビゲータなど、データベース内のオブジェクトを指定します。

アクション

Notes オブジェクトに対して実行する操作を特定します。 たとえば、?OpenDatabase、?OpenView、?OpenDocument、?OpenForm、?ReadForm、?EditDocument などがあります。 URL にどのアクションも指定されていない場合は、デフォルトが使用されます。

デフォルト： ?Open

引数

Domino サーバがどのようにオブジェクトを送るかを定義します。 たとえば、アクションと引数が ?OpenView&Expand=5 である場合、この引数は展開形式で表示する際の行数を指定しています。

financials.nsf という名前の Notes データベース内のビューにアクセスする URL の例を次に示します。

```
http://www.example.com/financials.nsf/reports?OpenView
```

Domino のエイリアス

Notes データベース規約の 1 つにオブジェクトのエイリアスの作成があります。たとえばエイリアスでは、オブジェクト名の代わりに Notes ID またはレプリカ ID を使ってリソースを識別できます。 エイリアスを使用すると、開発者のプログラミング作業が簡単になります。というのも、Notes リソースの名前を変更しても、コードを変更する必要がなくなるからです。

次の Domino URL はそれぞれ異なるエイリアスによって識別されていますが、すべて同一のリソースにアクセスします。

- http://www.domino.com/85255e01001356a8852554c20756?OpenView
- http://www.domino.com/85267E00075A80C/people?OpenView
- http://www.domino.com/__852567E00075A80C.nsf/people?OpenView

Domino Web エージェントはデータベースリソースの識別方法に関係なく、Domino 命名規約に従ったすべての ID を、リソースの名前に基づく標準の URL に変換します。これにより、SiteMinder ポリシー ストアへのデータ入力が簡略化されます。

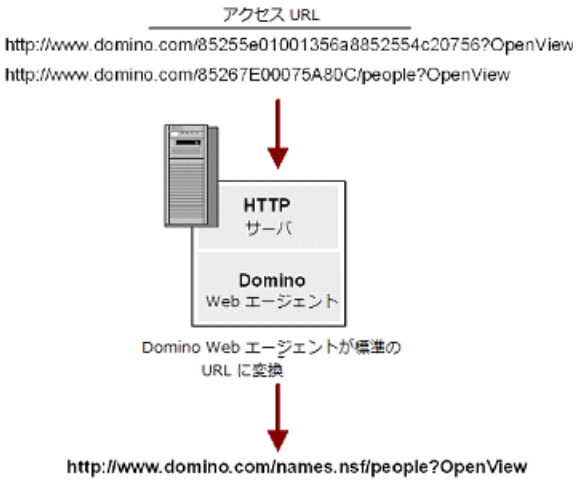
たとえば以下の Domino URL は、names.nsf データベース内の people ビューを指しています。データベースとビューはそれぞれ、レプリカ ID と Notes ID で参照されています。

- http://www.domino.com/85255e01001356a8852554c20756?OpenView
- http://www.domino.com/85267E00075A80C/people?OpenView

Domino Web エージェントはこれらの URL を次の標準 URL に変換します。

- http://www.domino.com/names.nsf/people?OpenView

以下の図に、エイリアスから名前付きオブジェクトへの変換を示します。



Domino Web エージェントの設定

Domino Web エージェントは、すべての Web エージェント標準設定を使って次のことを実行できます。

- ポリシー サーバと通信する Web エージェントの設定
- 仮想サーバのエージェント ID の追加と削除
- Web エージェントの設定の変更

- シングル サインオンの設定
- エラー メッセージ ロギングの設定

これらの設定は、ポリシー サーバ上で集中的に、またはエージェント設定ファイル内でローカルに実行することができます。

標準的な機能に加えて、設定可能な Domino 特有のパラメータもあります。

詳細情報

Domino 固有のエージェント機能の設定 (P. 385)

Domino 固有のエージェント機能の設定

Web エージェント標準設定のほかに、Domino Web エージェントの場合に限って設定可能な Domino 固有の設定パラメータが存在します。これらの設定により、Domino が SiteMinder と連携してユーザを認証して許可する方法が決まります。 この設定は、ポリシー サーバ上のエージェント設定オブジェクトに一元的に設定するか、または Web サーバ上のエージェント設定ファイルにローカルに設定できます。

注: Domino Web エージェントでは、ユーザ アクティビティの追跡に使用される監査機能はサポートされません。

omino に関するユーザ ディレクトリの指定

Domino ディレクトリはすべての Domino サーバと統合化されます。 Domino サーバの LDAP サービスを有効化すると、ポリシー サーバによるユーザ認証/許可時に Domino ディレクトリを使用できます。 Domino の LDAP サービスを有効化した場合、認証用に別のユーザ ディレクトリを設定する必要はありません。

LDAP サービスを有効化する方法については、Domino サーバのマニュアルを参照してください。

詳細情報:

CA への連絡先 (P. 3)

Domino サーバに関するポリシーを作成する場合のガイドライン

Domino サーバ用の SiteMinder ポリシーを作成する場合は、次のガイドラインに従ってください。

- ユーザは、親ドキュメントを持つフォームを開いてそのフォームのデフォルト値を参照することができます。 親ドキュメントとは、そのドキュメントの作成時に使用された元のフォームのことです。 無許可のユーザが、アクセス権のないフォームのデフォルト値を表示するのを防ぐには、SkipDominoAuth パラメータを no に設定します。
- データベースを同じコンピュータ上で複製した場合、各データベースを別々に保護するには、対応するルールも複製する必要があります。
- Domino エージェントが Notes ドキュメントのエイリアスをフォームに関連付けることができない場合、ドキュメントごとに異なるルールを指定してドキュメントを保護する必要があります。
- 特定の種類のデータベース ドキュメントの場合、Domino サーバは URL コマンド内で、$DefaultView、$DefaultForm、$DefaultNav、$SearchForm などの特殊な識別子を使用します。 Domino エージェントはこれらの識別子を標準の URL に変換してからドキュメントにアクセスします。

  $defaultNav の場合、Domino エージェントは ?OpenDatabase アクションを実行します。 これらの種類の識別子用に、追加のルールを作成する必要はありません。
- Notes データベース内のエイリアスは関連するリソースを保護します。 エイリアスが存在しない場合、リソース名またはコメントが関連するリソースを保護します。
- Lotus Notes ソフトウェアでは、種類の異なる複数のオブジェクトが、同一の名前またはエイリアスを持つことができます。 ?Open* などのように、?Open アクションでワイルドカードを使用するルールを作成した場合、このルールによって、エイリアスまたは名前を共有するさまざまな種類のリソースがすべて保護されることに注意してください。
- フォームは、それらのフォームにより作成されたドキュメントを保護します。 フォームで使用されるアクションは ?ReadForm です。
- Domino エージェントは拡張子 .nsf のファイルを保護します。 この拡張子を IgnoreExt パラメータに追加しないでください。

Domino のポリシーの設定

Domino サーバでは、同一の Notes オブジェクトをさまざまな方法を使って表現できます。オブジェクトの識別には、名前、レプリカ ID、ユニバーサル ID、およびエイリアスが使用できます。

Domino Web エージェントは、Domino サーバとの通信を効率的に行うために、Notes リソースへのアクセス リクエストを処理する際にオブジェクト名のみを使用します。これにより、SiteMinder ポリシー ストアはエントリを認識できるようになります。

任意のリソースへのアクセス方法を URL で表現すると、次のようになります。

http://host/database.nsf/resource_name?Open

Domino サーバ リソースのルールの作成

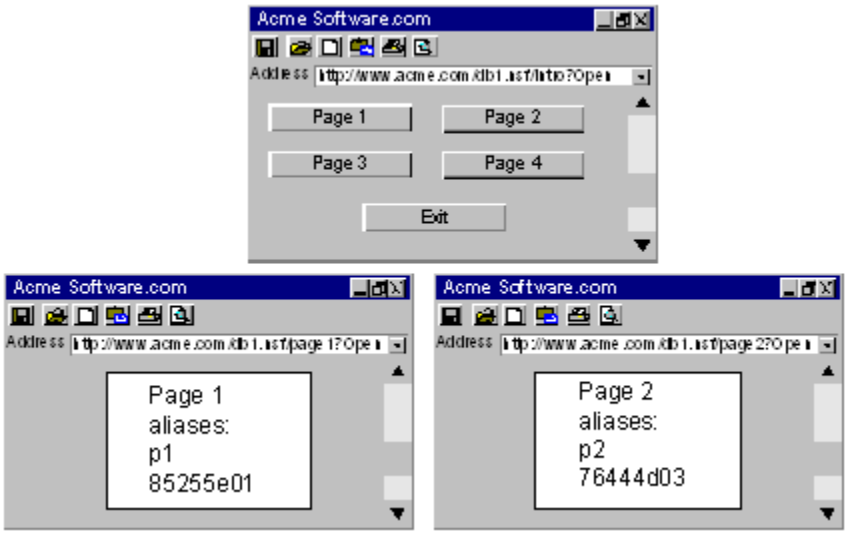
ルールを作成するには、Notes データベースリソースのアクションについて考慮する必要があります。 アクションが指定されていないリソースのデフォルトのアクションは ?Open になります。SiteMinder ポリシーに含まれるルールは、デフォルトのアクション ?Open、および ?OpenDatabase、?OpenView、?OpenDocument、?OpenFrameset など、?Open の同等のアクションを考慮する必要があります。

Domino Web エージェントを使用すると、ポリシー管理者は、同一リソースを参照する多くのエイリアスに対して 1 つのルールを作成することができます。 ルールを 1 つしか作成する必要がないのは、Domino エージェントは複数の Domino リソース表現を 1 つの URL に変換するためです。SiteMinder ポリシーのルールを作成する際は、Domino エージェントのこの機能を検討することが重要です。

レルムとルールの作成には、管理 UI を使用します。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

以下の図の URL は、db1.nsf という Notes データベースが存在する Acme の Domino サーバへのリンクです。 このデータベースには、2 つのファイル（ページ 1 とページ 2）が含まれています。



例 1： 1 つのドキュメントとそのすべてのエイリアスの保護

ページ 1 およびそのすべてのエイリアスへのアクセスに対して、レルム db1.nsf にルールを 1 つのみ作成します。 Domino エージェントは、異なるすべての命名規則を解釈して 1 つの標準 URL 形式に変換することができます。

レルムおよびルールについて、次の手順に従います。

- レルムの作成時に、ページ 1 が含まれているデータベースに以下のようにリソース フィルタを指定します。たとえば、データベース内のすべてのファイルを保護するには、以下のように設定します。

  Resource filter: /db1.nsf/

  ページ 1 に加えて、そのすべてのエイリアスを保護するには、以下のように設定します。

  Resource filter: /db1.nsf/page1

- ページ 1 の任意のアクションを保護するルールを作成するには、アスタリスク（*）を［ルールのプロパティ］ダイアログ ボックスの［リソース］フィールドに入力します。 例：

  Resource: *

  このワイルド カード * は、ポリシーに結び付けられているユーザが、ページ 1 に対して ?Open、?EditDocument など任意のアクションを実行できることを表します。

例 2： 同一データベース内の異なるドキュメントの保護

ページ 1 のほかに db1.nsf データベース内の ページ 2 を保護するには、以下のような 2 番目のルールを作成する必要があります。

Resource Filter: /db1.nsf/page2

Resource: *

例 3： 同一リソースに対する異なるアクションの保護

あるリソースのアクションを別々に保護するには、たとえば、特定のユーザだけにアクション ?EditDocument を実行させ、その他のユーザにアクション ?ReadForm を実行させる場合は、各リソースのアクションごとに異なるルールを次のように定義する必要があります。

- ルール 1

  Resource Filter: /db1.nsf/page1

Resource: ?OpenView

- ルール 2

  Resource Filter: /db1.nsf/page1

  Resource: ?EditDocument

また、次のように 1 つのルールを使用することもできます。

Resource Filter: /db1.nsf/page

Resource: ?Open*

注: ［リソース］フィールドでは、?Open の前にスラッシュ（/）を付けません。

このリソースのエイリアスが存在する場合でも、このルール 1 つでオリジナルのページとそのすべてのエイリアスが保護されます。

アクションごとにルールを作成する代わりに、次のように 1 つのルールを指定して、すべてのアクションをカバーするワイルドカードを使用することもできます。

Resource filter: /db1.nsf/page

Resource: ?Open*

このルールを使用すると、次のようなリソースを保護することになります。

http://www.acme.com/db1.nsf/page*?Open*

注: ルールをリテラルにするには、正規表現を記述してください。

Domino サーバによるユーザ認証

SiteMinder がすでにユーザの認証および許可を完了している場合でも、Domino サーバはユーザの認証および許可を行う必要があります。SiteMinder は Domino の認証プロセスと連携して Domino サーバにユーザ ID を提供します。この ID は、ユーザとその権限の一覧が保存された Domino ディレクトリ内にも設定されます。 Domino サーバはこの ID を使って、ユーザの認証およびデータベース リソースへのアクセス許可を行います。

注: ユーザ名は明確に解決される必要があります。そうしないと、Domino エージェントで認証リクエストが拒否されます。 その場合は、使用中のユーザ ディレクトリに調整を加える必要が生じることがあります。

Domino エージェントは、ユーザ ID を次のいずれかとして、Domino サーバに提供します。

- スーパーユーザ
- 実ユーザ
- デフォルト ユーザ

Domino サーバとの通信時に Domino Web エージェントが使用する ID を決定するには、次のパラメータを設定します。

SkipDominoAuth

サーバ認証のために Domino サーバに渡す名前を指定します。

DominoSuperUser

Domino サーバ上のすべてのリソースにアクセスできるユーザを指定します。

DominoDefaultUser

Notes データベースに対するデフォルトのアクセス権を持つユーザを指定します。これは、そのユーザが一般的なアクセス権限を持っていることを意味します。

注: DominoSuperUser と DominoDefaultUser は、ローカルのエージェント設定ファイルで設定することも、エージェント設定オブジェクトで一括設定することもできます。 エージェント設定ファイル内では、これらの設定項目の値は暗号化されています。 エージェント設定オブジェクト内では、それらの値を暗号化するか、プレーンテキストのままにするかを選択できます。

詳細情報

SiteMinder によるユーザ認証 (P. 396)
実ユーザまたはデフォルト ユーザとしての認証 (P. 393)
Domino スーパー ユーザとしての認証 (P. 392)

Domino スーパー ユーザとしての認証

Domino スーパー ユーザは、Domino サーバ上のすべてのリソースにアクセスできるユーザです。 Web サイトやポータルの設計で SiteMinder の使用が考慮されている場合は、SiteMinder ポリシーを実装することによって、リソースおよびアプリケーションを保護します。 その結果、Domino サーバが独自のセキュリティ機構を使ってユーザのアクセスを制限する必要性はなくなります。 この場合、ユーザを Domino の認証目的ではスーパー ユーザとして識別することができます。

ユーザをスーパー ユーザとして識別するには、SkipDominoAuth パラメータを有効化し、DominoSuperUser パラメータに値を指定します。 このアクションにより、Domino ではなく SiteMinder がユーザを認証するようになります。 指定したユーザは、Domino ディレクトリ内にも存在している必要があります。

実ユーザまたはデフォルト ユーザとしての認証

対象ユーザが Domino ディレクトリに定義されている場合は、Domino はそのユーザ名を使ってユーザ認証を行います。ただし、そのユーザが Domino ディレクトリに存在せず、SiteMinder によって別のユーザ ディレクトリに対して認証済みである場合は、Domino Web エージェントは Domino サーバに対してそのユーザを DominoDefaultUser として識別します。

デフォルト ユーザは Notes データベースに対するデフォルトのアクセス権を持ちます。つまりこのユーザは、ACL で設定されている Domino のディポジッタ、リーダ、作成者レベルのアクセス権などの一般アクセス権限を持つことになります。

Domino エージェントがこの値を使用するには、SkipDominoAuth パラメータに no を設定する必要があります。

SiteMinder で保護する必要のない Notes データベースが存在することがあります。 SiteMinder によって保護されないリソースは、デフォルトの Domino ユーザとして認証されません。 代わりに、Domino サーバは（匿名アクセスが無効である場合）、ユーザに認証情報を求めるプロンプトを表示します。

Domino デフォルト ユーザおよび Domino スーパー ユーザの変更

DominoDefaultUser および DominoSuperUser の各パラメータを変更するには、以下の作業のいずれかを実行します。

- ローカルで設定する場合は、エージェント設定オブジェクト内でこれらのパラメータを変更します。

  DominoDefaultUser および DominoSuperUser の各設定は、エージェント設定オブジェクト内で変更できます。 値を暗号化するか、プレーンテキストのままにするかを選択できます。

  注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

- encryptkey ツールを使用して、エージェント設定ファイル内のパラメータを変更します。

  エージェント設定ファイル内では、DominoDefaultUser および DominoSuperUser の各値を暗号化する必要があります。 したがって、encryptkey ツールを使用して、これらの値を変更する必要があります。

  重要: エージェント設定ファイルの中で、これらの設定項目を直接編集することは避けてください。

Encryptkey の使用による Domino デフォルト ユーザまたは Domino スーパー ユーザの設定

エージェント設定ファイル内で DominoSuperUser または DominoDefaultUser の値を設定または変更するには、以下の手順に従います。

1. 以下のいずれかを実行します。
   - UNIX: Domino エージェントの bin ディレクトリに移動します。例:

     /$HOME/ca/SiteMinder/Web Agent/bin
   - Windows: コマンド プロンプト ウィンドウを開き、Domino エージェントの Bin ディレクトリに移動します。 例:

     C:¥Program Files¥ca¥SiteMinder Web Agent¥Bin
2. 以下の引数を指定して、encryptkey ツールを実行します。
   - DominoSuperUser の場合:

     encryptkey -path *path_to_Agent_config_file*

     -dominoSuperUser *new_value*
   - DominoDefaultUser の場合:

     encryptkey -path *path_to_Agent_config_file*

     -dominoDefaultUser *new_value*

   例:

   encryptkey -path "c:¥program files¥ca¥SiteMinder Web Agent¥Bin¥Lotus Domino5¥webagent.conf"

   -dominoSuperUser admin

   注: エージェント設定ファイルのパスには、webagent.conf などのファイル名を含める必要があります。 また、パス内の任意の値にスペースが含まれている場合、パス全体を引用符で囲む必要があります。

   注: encryptkey ツールは、SiteMinder Web エージェント キットには含まれていませんが、 Domino ユーザに役立つツールです。Domino ユーザはこのツールを扱って、ローカル設定用の暗号化された DominoSuperUser 値を生成することができます。 このツールのダウンロードについては、サポートにお問い合わせください。

SiteMinder によるユーザ認証

Domino ではなく SiteMinder でユーザの認証を行うには、SkipDominoAuth パラメータを yes に設定します。

SkipDominoAuth に yes が設定され、スーパーユーザが定義されていると、最初に SiteMinder がユーザを識別して許可します。 次に、Domino Web エージェントがそのユーザをスーパーユーザとして Domino サーバに通知します。そのユーザはスーパーユーザなので、適切な ACL が割り当てられていることを前提として、Domino サーバ上のすべてのリソースにアクセスできます。

ユーザが Domino ディレクトリに保存されていない場合にも、SkipDominoAuth パラメータに yes を設定する必要があります。これは、許可権限に使用すべき ID が Domino 内に存在しないためです。

SkipDominoAuth を no に設定した場合、Domino は実ユーザ名またはデフォルト ユーザ名を使って独自にユーザを認証します。

以下の表は、SkipDominoAuth パラメータの設定がユーザの識別方法にどう影響するかを示しています。

SkipDominoAuth の値	Domino サーバでの識別の種類	説明
yes	スーパーユーザ	スーパーユーザは Domino ディレクトリに定義されている必要があります。
no	実ユーザ	ユーザは Domino ディレクトリに存在している必要があります。
no	デフォルト ユーザ	ユーザは Domino ディレクトリに存在している必要があります。
no	スーパーユーザ	要求されたリソースは自動的に許可されます。つまり、このユーザには認証チャレンジは表示されません。

詳細情報

実ユーザまたはデフォルト ユーザとしての認証 (P. 393)

SiteMinder ヘッダを使用した認証

DominoUseHeaderForLogin および DominoLookUpHeaderForLogin の各パラメータを使用して、Domino ユーザを認証することもできます。

DominoUseHeaderForLogin

SiteMinder のヘッダの値を Domino Web サーバに渡すように Domino Web エージェントに指示します。 Domino サーバはそのヘッダのデータを使用して、自らのユーザ ディレクトリの中に存在しているユーザを識別します。

このパラメータは、ヘッダ名と同じ値に設定します。 たとえば、DominoUseHeaderForLogin="HTTP_SM_USER" と指定した場合、Web エージェントはユーザのログイン名を Domino サーバに渡します。

DominoLookUpHeaderForLogin

ユーザがリソースへのアクセスを要求したときに、そのユーザが Domino ユーザ ディレクトリ内で一意か、またはあいまいかを、Domino Web サーバに問い合わせることを Domino Web エージェントに指示します。Jones という 1 人のユーザがリソースへのアクセスを試み、ユーザ ディレクトリ内に Jones というユーザが複数存在しているときに、このチェックは役立ちます。 このパラメータが no に設定されている場合、Domino Web エージェントは Domino Web サーバに確認しません。

デフォルト： Yes

Domino セッション認証の無効化

SiteMinder には、認証機能と許可機能が用意されているので、Domino のセッション認証機能は必要ありません。 Web エージェントがインストールされている場合は、このセッション機能を無効にする必要があります。

場合によって、Domino のセッション認証を有効にしておくと、ユーザ セッションが不正な動作を引き起こすことがあります。 この動作の変化は、SiteMinder 対応サイトのセキュリティには影響しません。 これは、SiteMinder と Domino のセッション管理ルールの AND 演算を反映しています。

Domino での匿名 SiteMinder 認証方式の使用

匿名 SiteMinder 認証方式を Domino エージェントで使用するには、以下のパラメータを設定します。

DominoUserForAnonAuth

匿名ユーザに対する値を指定します。 匿名 SiteMinder 認証方式で保護されている Domino リソースにユーザがアクセスするときに、この値が Domino サーバに送信されます。

デフォルト： no（匿名認証方式を使用 *しない*）

例： anonymous（匿名認証方式の場合に使用）

Domino で匿名の SiteMinder 認証方式を使用するときのみ、前のパラメータは適用されます。他の認証方式またはサーバ タイプ用の値は変更 *しません*。

認証を目的とした Domino エージェントによる認証情報の収集

認証情報コレクタは、Web エージェント内にあるアプリケーションの 1 つで、フォーム、SSL、Windows の各認証方式、および複数の cookie ドメインへのシングル サインオンに関して、ユーザ認証情報を収集します。 認証情報コレクタが収集する認証情報は、保護されたリソースからなる特定のグループに関して設定された認証方式のタイプを基礎としています。

Domino Web エージェントを認証情報コレクタとして機能させるには、エージェント設定ファイルの中でファイル拡張子として表現されているさまざまな MIME タイプを設定する必要があります。

認証情報コレクタは、一般的に自動認証されます。つまり、1 つのファイル拡張子をこれらのパラメータに追加した時点で、その拡張子はデフォルトで IgnoreExt パラメータの中に含まれます。 Domino サーバは、これらの拡張子がついたファイルを含む URL を正しく処理することができません。そのため、Domino エージェントはそのようなファイルを無視する必要があります。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

Domino Web エージェントによる FCC リダイレクト用 URL のマップ

フォーム認証方式を使用して Domino ビュー（.nsf）のリソースを保護するには、フォーム認証情報コレクタにリダイレクトする前に、URL をマップする必要があります。

次の手順に従ってください:

1. DominoNormalizeUrls パラメータの値を yes に設定します。
2. DominoMapUrlForRedirect パラメータの値を yes に設定します。

   Domino の URL は FCC へリダイレクトされる前にマップされます。

URL 正規化の無効化

URL 正規化の目的は、URL を Domino の表現から一般的なブラウザで使用される URL 形式に変更することです。 Domino Web エージェントは Domino Web サーバの API を利用して Domino の URL を正規化します。

正規化プロセスの際、Domino サーバの API は、正規化された URL に復帰文字（16 進数の 0x0D）または改行文字（16 進数の 0x0A）あるいはその両方を追加して URL を定期的に返します。 これらの文字の追加は、特定の Notes データベース（.nsf）ファイルおよびこれらのファイル内のアクセスパターンに関連していると考えられます。

以下の例に、復帰文字が追加された正規化後の URL を示します。

- URL：
  http://server.ca.com:80/agentrunner.nsf/be68f4545348400461332?OpenView
- URL のマップ先：
  http://server.ca.com:80/agentrunner.nsf/AgentContext?OpenView
- URL の正規化：
  http://xxxxx.ca.com:port/agentrunner.nsf/0x0d/AgentContext?OpenView

必要に応じて、以下のパラメータを使用して、Domino リソース ID を含む URL が正規化されないようにすることができます。

DominoNormalizeUrls

SiteMinder Web エージェントが、フォーム認証情報コレクタにリダイレクトする前に、Domino の URL を URL フレンドリ名に変換するかどうかを指定します。

Domino の URL を変換するためには、MapUrlsForRedirect パラメータも yes に設定する必要があります。

DominoNormalizeUrls パラメータが no の場合は、MapUrlsForRedirect パラメータが yes に設定されていても、URL は正規化されません。

重要：DominoNormalizeUrls パラメータを no に設定した場合、Notes データベース内の個々のドキュメントを保護することはできません。Domino Web サーバのデータベース全体またはサブディレクトリのみを保護することができます。

デフォルト： yes

正規化をオフにして URL が変更されないようにするには、DominoNormalizeUrls パラメータを no に設定します。

Lotus Notes ドキュメントへのアクセスの制御

Web エージェントでは、Domino 上の Lotus Notes ドキュメントの保護を詳細に制御することができます。この保護は、以下のパラメータで制御します。

DominoLegacyDocumentSupport

Web エージェントが Domino 環境内の保護されている Lotus Notes ドキュメントに対するユーザ要求を処理する方法を指定します。このパラメータを yes に設定すると、要求されたドキュメントに対してのみ、ユーザに ReadForm 許可が与えられます。

デフォルト： No

Notes ドキュメントにアクセスしているときにユーザが要求したアクションを処理するように Web エージェントを設定するには、DominoLegacyDocumentSupport パラメータを使用します。 この方法で、Domino の保護をより詳細に制御できます。

Notes ドキュメントには、個別の名前がありません。 それらのドキュメントは、作成の際に使用されたフォームへの参照と共に、データベース内に保存されています。ユーザが何らかの Notes ドキュメントをリクエストした場合、Domino Web エージェントはそのリクエストを URL へ変換することにより、該当するフォームを見つけます。 この URL の中に、Domino に対する元のアクションが含まれています。 フォームが見つからない場合は、何も使用されません。

例：

"http://server.domain.com/db.nsf?OpenDocument"

?OpenDocument や ?EditDocument など、ユーザがこの URL で指定されているドキュメントに対して要求した Domino アクションを Web エージェントが実行するようにするには、DominoLegacyDocumentSupport パラメータを no に設定します。

たとえば、次のような URL へのリクエストが発生したとします。

http://www.dominoserver.com/names.nsf/9348730948938987785784395880920398579834 9?EditDocument

Domino エージェントは、上記の URL を、次のように変換します。

http://www.dominoserver.com/names.nsf/Person?EditDocument

Person は、どのドキュメントを作成する際に使用されたフォームの名前であり、元の URL の中では、NotesID によって指定されています。

Notes ドキュメントにアクセスする際に、4.6 より前の動作を実行するよう Domino Web エージェントに指示するには、このパラメータを yes に設定します。これは、?ReadForm アクションのみを許可することを意味します。レガシー ドキュメントサポートが有効になっている場合、Domino エージェントは上記の例の URL を、次のように変換します。

http://www.dominoserver.com/names.nsf/Person?ReadForm

Notes ドキュメント名の変換

ビューやフォームと異なり、Notes ドキュメントは名前を持ちません。Notes ドキュメントは、ドキュメント作成時に使用したフォームのリファレンスと共にデータベースに保存されます。 ユーザがドキュメントにアクセスしようとしたときに、Domino Web エージェントがそのドキュメントを読み取り可能な名前に変換できなかった場合、エージェントはそのドキュメントを生成したフォームの名前を使って URL を作成します。 ただし、この規則はドキュメントにのみ適用されます。 元のフォームが存在しない場合、エージェントは埋め込まれたフォームを使用します。 どちらも存在しない場合、Domino の識別子 $defaultForm によってドキュメントが保護されます。

たとえば、次のような URL を受信したとします。

http://www.domino.com/names.nsf/8567489d60034we50938450098?OpenDocument

この場合、エージェントは次の URL を使用します。

http://www.domino.com/names.nsf/Person?ReadForm

この例で、Person はドキュメントの名前です。

Domino エージェントの完全ログオフ サポートの設定

完全ログオフ機能では、以下のパラメータで作成するカスタム ログアウト ページが使用されます。

LogOffUri

カスタム Web ページの URI を指定して完全なログアウト機能を有効にします。 ユーザが正常にログオフした後にこのカスタム Web ページが表示されます。 ブラウザ キャッシュ内に格納できないようにこのページを設定します。設定しなかった場合、ブラウザは、ユーザをログ オフせずに、そのキャッシュからログアウト ページを表示する場合があります。 この状況が発生すると、不正なユーザにセッションの支配権を握る機会を与える可能性があります。

注: CookiePath パラメータが設定されているときは、LogOffUri パラメータの値が同じ cookie パスを指している必要があります。 たとえば、CookiePath パラメータの値が example.com に設定されている場合、LogOffUri は example.com/logoff.html を指している必要があります。

デフォルト：（CA SiteMinder Agent for SharePoint r12.0.3.0 *以外*のすべてのエージェント）デフォルトなし

制限： 複数の URI 値を指定できます。 完全修飾 URL は使用 *しないでください*。*相対* URI を使用します。

例： （CA SiteMinder Agent for SharePoint r12.0.3.0 *以外*のすべてのエージェント） /Web pages/logoff.html

次の手順に従ってください:

1. ユーザのログオフ用のカスタム HTTP アプリケーションを作成します。たとえば、ユーザを指定した URL にリダイレクトするための終了ボタンまたはサインオフ ボタンを追加します。
2. ログアウト ページを Web ブラウザにキャッシュできないように設定します。 この設定により、ページはブラウザのキャッシュではなく Web サーバから常に提供されるので、セキュリティが向上します。 たとえば、HTML ページの場合は、ページに次のようなメタタグを追加します。

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="Expires" CONTENT="-1">
```

重要: 一部の Web ブラウザは、メタ タグをサポート *していません*。代わりに、キャッシュ コントロール HTTP ヘッダを使用してください。

3. 以下の手順で LogOffUri パラメータを設定します。
   a. 必要に応じて、ポンド記号（#）を削除します。
   b. ユーザをログオフするカスタム HTTP ファイルの URI を入力します。 完全修飾 URL は使用 *しないでください*。

      完全なログアウト機能が設定されます。

詳細情報

完全ログオフの仕組み (P. 295)

Domino Web エージェントと WebSphere Application Server の連動

Domino Web サーバは、リクエストが WebSphere サーバに転送される前に、それらのリクエストをインターセプトするフィルタ プラグインを提供することにより、WebSphere Application Server のフロント エンドとして動作します。

Domino サーバによる、保護されていない SiteMinder リソースの認証

SiteMinder で保護しないリソースが Domino サーバ上にあるとします。 それらのリソースも、代わりに Domino サーバで保護できます。 これらのリソースを保護するには、以下のパラメータを設定します。

UseDominoUserForUnprotected

Domino のサーバのみが保護する（SiteMinder は保護していない）リソースに対する Domino ユーザによる要求を、Domino サーバが認証するかどうかを指定します。

このパラメータの値が yes の場合、エージェントは Domino ユーザを Domino サーバへ渡します。 Domino サーバはユーザを認証します。このパラメータの値が No の場合（またはパラメータが無効な場合）、エージェントは Domino ユーザを Domino サーバへ渡しません。 Domino サーバはユーザを認証しません。

デフォルト： disabled

次の手順に従ってください:

1. 前のパラメータを見つけます。
2. パラメータの前の # （コメント）文字を削除します。
3. パラメータの値を yes に変更します。

後方互換性の設定

SiteMinder エージェント の後方比較性を管理するために、以下の設定項目のうち、必要なものを設定します。

- レガシー URL エンコードを受け入れます (P. 407)。
- コンテンツ タイプが POST 要求でどのように転送されるか決定します (P. 372)。
- HOST ヘッダを送信しないテスティング ツールに対応します (P. 408)。

レガシー URL エンコードの受け入れ

CA によって使用されるレガシー URL エンコーディングでは、ドル記号 ($) 文字を使用します。 ドル記号が問題を引き起こす場合、以下のパラメータを使用して、Web エージェントにドル記号の代わりにハイフン（-）文字を使用させることができます。

LegacyEncoding

Web エージェントで、レガシー URL 内のすべてのドル記号（$）文字を強制的にハイフン（-）に置換します。 これにより、MSR、パスワード サービス、および DMS に対する下位互換性も保証されます。このパラメータを no に設定すると、Web エージェントは文字列の SM を -SM- に変換します。 このパラメータを yes に設定すると、Web エージェントはドル記号（$）文字を変換しません。

デフォルト：（フレームワーク エージェント） No

デフォルト：（従来のエージェント） Yes

ドル記号の代わりにハイフンを使用してレガシー URL をエンコードするには、LegacyEncoding パラメータの値を no に設定します。

POST 要求でのコンテンツ タイプの転送方法の選択

Apache Web サーバを使用している場合、POST リクエストの間にコンテンツがサーバに転送される方法を、以下のパラメータで制御できます。

LegacyStreamingBehavior

コンテンツが POST 要求中にサーバにどのように転送されるかを指定します。 このパラメータの値が yes に設定されると、以下を*除く*すべてのコンテンツ タイプはストリームになります。

- text/xml
- application/x-www-form-urlencoded

このパラメータの値が no に設定されると、すべてのコンテンツ タイプがスプールされます。

デフォルト： No

POST 要求でのほとんどのタイプのコンテンツをストリームするには、LegacyStreamingBehavior パラメータの値を［はい］に変更します。

HOST ヘッダを送信しないテスティング ツールへの対応

SiteMinder Web エージェントでは、HTTP リクエスト内の HOST ヘッダの値を使用して以下の設定を判断します。

- エージェント名
- Server name
- サーバの IP アドレス

HTTP バージョン 0.9 および 1.0 では HOST ヘッダを使用しないため、SiteMinder Web エージェントでは HTTP バージョン 1.1 リクエストのみを受け入れます。 そのため、HOST ヘッダを送信しないテスト ツールで問題が生じています。Web エージェントでそれらのリクエストが拒否されるからです。

SiteMinder 12.52 では、HOST ヘッダ値を定義するための新しいエージェント設定パラメータがサポートされます。 Web エージェントでは、HOST ヘッダが含まれないすべてのリクエストでこの値を使用します。

HOST ヘッダを送信しないテスティング ツールに対応する方法

1. 以下のいずれかのアイテムを開きます。
   - 中央設定を使用している場合は、エージェント設定オブジェクトを開きます。
   - ローカル設定を使用している場合は、LocalConfig.conf ファイルを開きます。
2. 以下のパラメータを追加します。

   DefaultHostName

   HOST ヘッダに対する値を定義します。HTTP バージョン 0.9 または 1.0 リクエスト（HOST ヘッダなし）を送信するテスト/パフォーマンス ツールを使用するには、エージェント設定オブジェクトまたは LocalConfig.conf ファイルにこのパラメータを追加します。 このパラメータが設定されていない場合、Web エージェントでは HTTP 1.1 リクエストのみを受け入れます。

   デフォルト： なし（空白）

   例： webserver.example.com
3. 上記のパラメータの値を適切なホスト名に設定します。 前述の例を参照してください。

4. 以下のいずれかのアイテムを保存して閉じます。
   - 中央設定を使用している場合は、エージェント設定オブジェクトを保存して閉じます。
   - ローカル設定を使用している場合は、LocalConfig.conf ファイルを保存して閉じます。

   Web エージェントでは、HOST ヘッダのない HTTP リクエストに対して代わりに DefaultHostName の値を使用します。

フェデレーションドメイン用のエージェントの設定

SiteMinder が レガシー フェデレーション SP として動作している場合、SAML 2.0 トランザクション用に Identity Provider Discovery （IPD）プロファイルを設定できます。 IPD によって、認証リクエストに対してどの IdP がアサーションを生成するかをユーザが選択できるようになります。

検出プロセスで、悪意のある Web サイトにユーザがリダイレクトされるのを防ぐことができます。認証リクエストを満たす IdP のドメインが検証されるよう Web エージェントを設定します。

検証プロセスを有効にするには、以下のパラメータの値を設定します。

ValidFedTargetDomain

（Federation のみ-SAML 2.0） Identity Provider Discovery を実装した場合に、フェデレーション環境の有効なドメインをすべてリスト表示します。

SiteMinder Identity Provider Discovery （IPD）サービスでリクエストを受信すると、リクエストの IPDTarget クエリ パラメータを調べます。 このクエリ パラメータは、Discovery サービスでリクエストを処理した後にリダイレクトする URL をリスト表示します。IdP の場合、IPDTarget は SAML 2.0 シングル サインオン サービスです。SP の場合、ターゲットは共通ドメイン cookie を使用するリクエスト アプリケーションです。

フェデレーション Web サービスでは、IPDTarget URL のドメインを、ValidFedTargetDomain パラメータに指定されたドメインのリストと比較します。URL ドメインが ValidFedTargetDomain に設定されたドメインの 1 つと一致する場合、IPD サービスは IPDTarget パラメータに示された URL にユーザをリダイレクトします。 このリダイレクトは SP の URL に対して行われます。

ドメインが一致しない場合、IPD サービスはユーザ リクエストを拒否し、ブラウザに 403 Forbidden が返されます。 また、FWS トレース ログおよび affwebservices ログにエラーが報告されます。 これらのメッセージは、IPDTarget のドメインが有効なフェデレーション ターゲット ドメインとして定義されないことを示します。

ValidFedTargetDomain を設定しない場合、検証は行われず、ユーザはターゲット URL にリダイレクトされます。

制限： フェデレーション ネットワーク内の有効なドメイン

デフォルト： デフォルトなし

ValidFedTargetDomain パラメータに有効なドメインを指定します。 この設定は複数パラメータなので、複数のドメインを入力できます。

ローカル設定ファイルを変更している場合は、たとえば以下のように、ドメインを別々にリスト表示します。

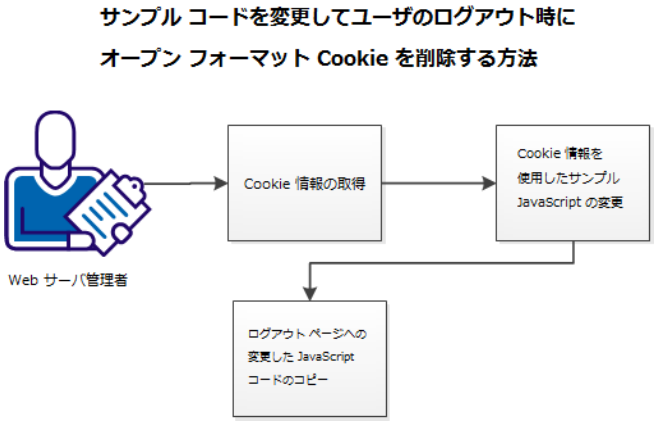
validfedtargetdomain=".examplesite.com"

validfedtargetdomain=".abccompany.com"

Identity Provider Discovery プロファイルの詳細については、「Federation Security Services Guide」を参照してください。

サンプル コードを変更してユーザのログアウト時にオープン フォーマット Cookie を削除する方法

SiteMinder はオープン フォーマット Cookie を認識、処理、削除*しません*。ユーザがログアウトした際にオープン フォーマット Cookie を削除する独自のクライアント側スクリプトを作成します。



次の手順に従ってください:

1. Cookie 情報を取得します (P. 412)。
2. Cookie 情報を使用してサンプル JavaScript コードを変更します (P. 413)。
3. 変更した JavaScript コードをログアウト ページにコピーします (P. 414)。

Cookie 情報を取得します

クライアント側のログアウト スクリプトは、ユーザのオープン形式 Cookie に関する以下の情報を必要とします。

- オープン形式 Cookie の名前（たとえば SMOFC）
- Cookie のパス（たとえば、/ [スラッシュは、オープン形式 Cookie のルート ディレクトリを示す]）
- Cookie が作成されるドメイン（たとえば、.example.com）

エージェント所有者または Web サーバ管理者から、この情報を取得します。

Cookie 情報を使用したサンプル JavaScript コードの変更

ユーザのオープン形式 Cookie に関する情報を取得した後にサンプルの JavaScript コードを変更します。

次の手順に従ってください:

1. テキスト エディタに以下のサンプル JavaScript コードをコピーします。

```
<html>
    <head>
        <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
        <META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
        <META HTTP-EQUIV="Expires" CONTENT="-1">

        <!-- ブラウザから Cookie を削除する JavaScript -->
        <script>
            // This function takes the cookie name, path and domain
            // and constructs a expired cookie so that the browser removes the
cookie from its store
            function eraseCookie(name, path, domain)
            {
                if (name)
                {
                  var delCookie = name + '=; expires=Thu, 01-Jan-70 00:00:01
GMT';
                  if (path && path.length > 0) delCookie += ';path=' + path;
                  if (domain && domain.length > 0) delCookie += ';domain=' +
domain;
                  document.cookie = delCookie;
                }
            }

            function showCookie(name)
            {
                var ckVal = null;
                var tC = document.cookie.split('; ');
                for (var i = tC.length - 1; i >= 0; i--)
                {
                    var x = tC[i].split('=');

                    if (name == x[0] && x[1])
                    {
                        ckVal = unescape(x[1]);
                        break;
                    }
                }

                if (ckVal)
```

```
                alert( name + ' = ' + ckVal);
            else
                alert('Cookie ' + name + ' does not exist');
        }
    </script>
</head>

<body>
    <p><a href="javascript:showCookie('SMOFC')" class="page">Click to show
Open Format Cookie</a><br />
    <p><a href="javascript:eraseCookie('SMOFC', '/', 'example.com')"
class="page">Click to remove Open Format Cookie</a><br />
</body>
</html>
```

2. 以下の表に示すように、すべてのデフォルト値を置換します。

これらのデフォルト値をすべて置換します。	オープン形式 Cookie のこれらの値を使用	この例を参考にしてください。
名前	オープン形式 Cookie の名前	SMOFC
パス	オープン形式 Cookie のパス	¥
ドメイン	オープン形式 Cookie のドメイン	example.com

3. サンプル JavaScript に環境で必要なその他の変更を加えます。
4. 変更された JavaScript を保存し、テキスト エディタを閉じます。

ログアウト ページへの変更した JavaScript コードのコピー

変更した JavaScript コードでログアウト ページを更新します。このコードは、ユーザがログアウトするとオープン形式 Cookie を削除します。

次の手順に従ってください:

1. テキスト エディタを使用して、Web サーバのログアウト ページを開きます。
2. 変更した JavaScript コードをログアウト ページにコピーします。
3. ページへの変更を保存し、テキスト エディタを閉じます。
4. 各 Web サーバに対して手順 1 ～ 3 を繰り返します。

第 18 章: パフォーマンス

保存された認証情報のタイムアウトの設定

ユーザが認証情報を保存することを選択した場合、ポリシー サーバは、そのユーザの認証情報を保管する永続的な cookie を作成することを Web エージェントに指示します。この cookie を使用することで、Web エージェントは、ユーザに認証情報を再要求する代わりに、cookie に保存されている認証情報に基づいてユーザを認証します。 永続的な cookie の保存期間は、以下のパラメータを使用して制御できます。

SaveCredsTimeout

ユーザ認証情報が含まれている永続的な cookie が保存される時間数を指定します。 この時間中に、Web エージェントは、cookie 内に保存されたデータでユーザを認証します。この時間を過ぎると、cookie は削除され、Web エージェントは再度ユーザ認証を試みます。

デフォルト： 720（30 日）

保存された認証情報のタイムアウトを設定するには、SaveCredsTimeout パラメータに目的の時間数を入力します。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

Web エージェント キャッシュ

Web エージェントは、ユーザ セッションとリソース情報をキャッシュメモリに格納します。 この手法は、Web エージェントの効率を向上させます。ユーザがアクセスを要求するたびに、Web エージェントがポリシー サーバから情報を取得する必要がなくなるからです。

キャッシュを設定することで、これらの情報の格納方法を管理できます。キャッシュ内のエントリ数は、キャッシュのサイズを決定します。 それぞれのキャッシュの総エントリ数は、指定された最大キャッシュ サイズを超えることはできません。

注: Web エージェント キャッシュ設定の変更を有効にするには、Web サーバを再起動します。

キャッシュ管理には、次のガイドラインが適用されます。

- キャッシュが満杯になると、最も直近で利用されていないエントリから新しいエントリに置き換えられます。
- リソース キャッシュの場合、ResourceCacheTimeout パラメータの値に達すると、エントリが削除されます。
- ユーザ セッション キャッシュの場合、レルムごとに設定したセッションのタイムアウト値に基づいてエントリが削除されます。

ポリシーを変更すると、SiteMinder はキャッシュ内のリソース情報を削除します。 また、管理 UI を使用して、ユーザ キャッシュとリソース キャッシュを手動で消去することもできます。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

エージェントのキャッシュを管理するために以下のパラメータを使用します。

- 匿名ユーザをキャッシングします (P. 417)。
- 最大のリソース キャッシュのサイズを設定します (P. 418)。
- 最大のユーザ セッション キャッシュのサイズを設定します (P. 419)。
- リソース エントリをキャッシュに保存しておく時間を制御します (P. 420)。
- リソース キャッシュを無効化します (P. 420)。

匿名ユーザのキャッシング

以下のパラメータを使用して、キャッシュに匿名ユーザの情報を格納するように Web エージェントを設定することができます。

CacheAnonymous

Web エージェントが匿名のユーザ情報をキャッシュするかどうかを指定します。 このパラメータは、たとえば以下の状況に対して設定できます。

- Web サイトのユーザのほとんどが匿名ユーザで、それらのユーザのセッション情報を格納したい場合。
- 登録ユーザと匿名ユーザの両方が Web サイトにアクセスする場合。

  匿名ユーザの情報のみでキャッシュが満杯になり、登録ユーザ用の領域がなくなってしまう可能性がある場合は、このパラメータを無効にすることをお勧めします。

デフォルト： No

キャッシュに匿名ユーザの情報を格納するには、CacheAnonymous パラメータの値を yes に設定します。

リソース キャッシュの最大サイズの設定

以下のパラメータを使用して、Web ページなど、Web エージェントが追跡するリソース キャッシュ エントリの最大数を設定することができます。

MaxResourceCacheSize

Web エージェントがそのリソース キャッシュ内で保持するエントリの最大数を指定します。エントリには以下の情報が含まれます。

- リソースが保護されるかどうかに関するポリシー サーバのレスポンス
- レスポンスで返される追加属性

最大値に達すると、新しいリソース レコードが最も古いリソース レコードと置き換わります。

これらをより大きな数値に設定する場合は、十分なシステム メモリがあることを確認してください。

OneView モニタを使用して Web エージェント統計を表示している場合は、ResourceCacheCount に表示される値が MaxResourceCacheSize パラメータで指定された値より大きいことがあります。これはエラーではありません。Web エージェントは、MaxResourceCacheSize パラメータを 1 つのガイドラインとして使用します。また、値は状況により異なります。これは、MaxResourceCacheSize パラメータはリソース キャッシュ内の平均サイズのエントリの最大数を示すためです。 実際のキャッシュ エントリは、あらかじめ識別された平均サイズより大きかったり小さかったりする可能性があります。したがって、実際の最大エントリ数は指定された値より多い場合や少ない場合があります。

注: フレームワーク エージェントなど、共有メモリを使用する Web エージェントの場合、キャッシュは MaxResourceCacheSize の値に基づいて一定サイズが事前に割り当てられ、それより増えることはありません。

デフォルト： （Domino Web サーバ） 1000

デフォルト： （IIS および Sun Java System Web サーバ） 700

デフォルト： （Apache Web サーバ） 750

リソース キャッシュの最大サイズを設定するには、以下の手順に従います。

1. MaxResourceCacheSize パラメータの値を、目的のリソース最大数に設定します。

2. フレームワーク エージェントでは、変更を適用するために Web サーバを再起動する必要があります。

   リソース キャッシュの最大サイズが変更されます。

ユーザ セッション キャッシュの最大サイズの設定

以下のパラメータを使用して、エージェントがセッション キャッシュ内で保持するユーザの最大数を設定することができます。

MaxSessionCacheSize

エージェントがそのセッション キャッシュ内で保持するユーザの最大数を指定します。 セッション キャッシュには、認証するユーザのセッション ID が正常に格納されます。 それらのユーザが同じセッション中に同じレルム内の別のリソースにアクセスした場合、エージェントはポリシー サーバをコールする代わりにセッション キャッシュの情報を使用します。この最大数に達すると、エージェントは最も古いユーザ レコードを新しいユーザ レコードと置き換えます。

このパラメータの値は、持続期間にリソースにアクセスしてそれを使用する予定のユーザの数に基づいて設定します。 これらをより大きな数値に設定する場合は、十分なシステム メモリがあることを確認してください。

デフォルト： （Domino Web サーバ） 1000

デフォルト： （IIS および Oracle iPlanet Web サーバ） 700

デフォルト： （Apache Web サーバ） 750

ユーザ セッション キャッシュの最大サイズを設定する方法

1. MaxSessionCacheSize パラメータの値を必要なユーザの最大数に設定します。
2. フレームワーク エージェントでは、変更を適用するために Web サーバを再起動する必要があります。

   ユーザ セッション キャッシュの最大サイズが変更されました。

リソース エントリをキャッシュに保存しておく時間の制御

以下のパラメータを使用して、リソース エントリをキャッシュに保存しておく時間の長さを変更することができます。

ResourceCacheTimeout

リソース エントリがキャッシュに保存される秒数を指定します。時間間隔の値を超えると、Web エージェントはキャッシュされたエントリを削除します。その後、保護されているリソースにユーザがアクセスしようとすると、Web エージェントはポリシー サーバに問い合わせます。

デフォルト： 600（10 分）

注: このパラメータの値を変更した場合は、変更を適用するために Web サーバを再起動する必要があります。

リソース エントリをキャッシュに保存しておく時間を変更するには、ResourceCacheTimeout パラメータを目的の秒数に設定します。

リソース キャッシュの無効化

動的な一意の URL を使用するアプリケーションを保護している場合は、リソース キャッシュを無効にすることをお勧めします。 アプリケーションによって使用される URL が一意であるため、それらはキャッシュから読み取られません。

リソース キャッシュを無効にするには、MaxResourceCacheSize の値をゼロに変更します。

Web エージェントの監視

エージェントのパフォーマンスを監視するために、以下の方法のうち必要なものを使用します。

- OneView モニタにより Web エージェントを監視します (P. 421)。
- CA Wily Introscope を使用してエージェントを監視します (P. 422)。

詳細情報:

Web エージェントとポリシー サーバ間の通信を管理する方法 (P. 64)

OneView モニタによる Web エージェントの監視

SiteMinder OneView モニタは、キャッシュ統計情報と他の情報をポリシー サーバへ送信します。管理者はポリシー サーバを使用して、Web エージェントを分析し、微調整することができます。 以下のパラメータを使用して SiteMinder OneView モニタを制御します。

EnableMonitoring

SiteMinder Web エージェントが監視情報をポリシー サーバに送信するかどうかを指定します。

デフォルト： No

Web エージェントで SiteMinder OneView モニタが使用されるようにするには、EnableMonitoring パラメータを yes に設定します。

注: 詳細については、ポリシー サーバ ドキュメントを参照してください。

CA Wily Introscope を使用した Web エージェントの監視

すでに CA Wily Introscope を使用している場合は、以下のパラメータを使用して SiteMinder Web エージェントの稼働状況を監視できます。

EnableIntroscopeApiSupport

SiteMinder Web エージェントに関する情報を収集し、プラグインを使用して CA Wily Introscope に送ります。 このパラメータは以下の設定を使用します。

- yes に設定されたとき、Wily プラグインは、データを収集するために API をコールします。
- no に設定されたとき、Wily プラグインはデータを備えた HTTP ヘッダを作成します。
- both に設定されたとき、Wily プラグインは API をコールし、かつデータを備えた HTTP ヘッダを作成します。
- none に設定されたとき、データは収集されません。

デフォルト： no

制限： yes、both、no、none

例： （HTTP ヘッダ） sm-wa-perf-counters = *server_name*.example.com:6180,86117203,86118343,1,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1125,0,15,1,1,750,750,

CA Wily Introscope を使用して Web エージェントの稼働状況を監視するには、EnableIntroscopeApiSupport パラメータの値を以下のいずれかに設定します。

- はい
- 両方
- いいえ

保護されていないリソースを無視します。

保護しないリソースに対する要求を無視することにより、SiteMinder のパフォーマンスを改善できます。 以下のパラメータが使用可能です。

- 特定のファイル拡張子を無視することによりオーバーヘッドを削減します (P. 424)。
- エージェントがどの仮想サーバを無視するかを指定します (P. 190)。
- URLs 内のクエリ データを無視します (P. 428)。
- URI への無制限のアクセスを許可します (P. 430)。

保護されていないリソースのファイル拡張子を無視することによるオーバーヘッドの削減

以下のパラメータを使用して、特定のタイプのリソースの要求を無視するように Web エージェントに指示することにより、SiteMinder のオーバーヘッドを縮小できます。

IgnoreExt

Web エージェントが SiteMinder ポリシーを確認せずに Web サーバに要求を渡すリソースのタイプを指定します。SiteMinder ポリシーによって保護されるレルムにアイテムが存在する場合でも、Web エージェントはこのパラメータによって指定されたそのアイテムへのアクセスを許可します。

以下の条件のどちらかを満たすリソースに対する要求を無視することができます。

- リソースが、Web エージェントに対して無視するよう指定した拡張子で終わっている場合。
- 保護されているリソースを表す URI にピリオド（.）が 1 つだけ含まれている場合。

  たとえば、要求されたリソースの URI が /my.dir/ である場合、Web エージェントは要求を直接 Web サーバへ渡します。

デフォルト： .class、.gif、.jpg、.jpeg、.png、.fcc、.scc、.sfcc、.ccc、.ntc

重要: IgnoreExt パラメータを設定する場合は注意してください。セキュリティの問題には、検討が必要なものがいくつかあります。

デフォルトでは、エージェントは、スラッシュ（/）で区切られた複数のピリオドを含むリソースに対する要求を*無視しません*。 Web エージェントは、以下の例に示された手順に従って、リソースの要求を処理します。

1. 拡張子 .gif が IgnoreExt パラメータに追加されます。 拡張子が .gif のリソースの要求は、Web エージェントによって無視されます。
2. 要求は以下の URI に対して行われます。

   /dir1/app.pl/file1.gif,
3. 一部の Web サーバが file1.gif リソースにサービスを提供する代わりにアプリケーションとして /dir1/app.pl を実行するので、Web エージェントはポリシー サーバに対して /dir1/app.pl/file1.gif をチェックします。

   Web サーバに問い合わせずに /dir1/app.pl/file1.gif へのアクセスを付与することにより、セキュリティ違反が発生した可能性があります。

保護されていないリソースのファイル拡張子を無視することによってオーバーヘッドを削減するには、IgnoreExt パラメータの値に無視するリソースの拡張子を追加します。

Web エージェントで無視する仮想サーバの指定

使用中のサイト内にある 1 台の Web サーバが複数の仮想サーバをサポートしている場合、それらの仮想サーバ上には、Web エージェントで保護したくないリソースが存在している可能性があります。 Web サーバ コンテンツのうち、どの部分を保護する必要があるのか Web エージェントが簡単に識別できるように、以下のパラメータを使用します。

IgnoreHost

Web エージェントで無視するあらゆる仮想サーバの完全修飾ドメイン名を指定します。 そのような仮想サーバ上にあるリソースは自動許可され、どのクライアントが要求を行ったかにかかわらず、Web エージェントは常にそれらのリソースへのアクセスを許可します。 許可は、ポリシーではなく Web エージェントの設定に基づいて決定されます。

IgnoreExt や IgnoreURL の各設定など、自動許可に関する他の項目より先に、無視されるホストに関する上記のリストが最初にチェックされます。 したがって、無視されるホスト上にあるリソースに関しては、ダブルドット ルールがポリシー サーバに対する許可の呼び出しをトリガすることはありません。しかし、拡張子に関するルールがそのようなリソースを無視することはありません。

IgnoreHost パラメータに対する URL エントリのホスト部分は、Web エージェントが読み取る、要求されたリソースのホスト ヘッダと完全に一致する必要があります。

注: この値では、大文字と小文字が区別されます。

URL で特定のポートを使用する場合、ポートを指定する必要があります。

一元管理されたエージェントでは、いくつかのサーバを表わすためにエージェント設定オブジェクトで複数値パラメータを使用します。ローカル設定ファイルで設定されたエージェントでは、ファイル内の個別の行に各ホストを列挙します。

例： （指定したポートと共に表示される URL）

IgnoreHost="myserver.example.org:8080"

例： （ローカル設定ファイル）

IgnoreHost="my.host.com"

IgnoreHost="your.host.com"

デフォルト： デフォルトなし

Web エージェントで無視する仮想サーバを指定するには、次のいずれかのタスクを行います。

- 中央設定では、無視するサーバをエージェント設定オブジェクトに追加します。 サーバが複数ある場合は、パラメータに複数値の設定を使用します。
- ローカル設定では、ローカル設定ファイルでサーバごとに個別の行を追加します。

指定された URL を使用するリソースは、Web エージェントによって無視され、それらのリソースへのアクセスが自動的に付与されます。

詳細情報

複雑な URI の処理 (P. 119)

URL 内のクエリ データの無視

IgnoreQueryData パラメータは、Web エージェントが URL を取り扱う方法に影響を及ぼします。 Web エージェントが URL 全体をキャッシュに格納せず、ルール処理のためにクエリ文字列と一緒に URI をポリシー サーバに送信するように設定すると、パフォーマンスが向上します。この場合は、以下のパラメータを使用します。

IgnoreQueryData

Web エージェントが URL 全体（クエリ文字列を含む）をキャッシュに保管し、ルール処理のために URI 全体をポリシー サーバに送信するかどうかを指定します。 完全な URL 文字列には、以下の例に示すように、URI、フック（?）、およびクエリ データが含まれます。

URI?*query_data*

デフォルトでは、リクエストの対象となった URL がキャッシュに保管されます。後続のリクエストでは、一致する URL がキャッシュで検索されます。 リクエスト内で URI が同一でもクエリ データが異なると、一致は失敗します。 クエリ データを無視すると、パフォーマンスが向上します。

IgnoreQueryData パラメータが yes の場合は、以下の処理が発生します。

- URL はフックの箇所で切り捨てられます。 URI だけがキャッシュされ、ポリシー サーバへ送信されます。クエリ データは、リダイレクトの適正な状態を維持するために、他の場所で維持されます。
- フック（?）の前にある部分のみがポリシー サーバに送信されて、ルールの処理が行われます。
- 以下の例に示す 2 つの URI は、同一のリソースとして処理されます。

  /myapp?data=1

  /myapp?data=2

IgnoreQueryData パラメータが no の場合は、以下の処理が発生します。

- その場合、URL 全体がキャッシュされます。
- URI 全体がポリシー サーバに送信されてルールの処理が行われます。

- 以下の例に示す URI は、異なるリソースとして処理されます。

  /myapp?data=1

  /myapp?data=2

デフォルト： No

Web エージェントが、処理のためにポリシー サーバに URI のみを送信するようにするには、IgnoreQueryData パラメータの値を yes に設定します。

重要: URL クエリ データに依存するポリシーがある場合は、この設定を有効にしないでください。

URI への無制限のアクセスの許可

SiteMinder で保護しない URI がある場合は、以下のパラメータを設定することによって、それらの URI への無制限のアクセスを無視して許可するように Web エージェントに命令できます。

IgnoreUrl

保護されていない URL 内の URI を指定します。URI と関連付けられたリソースにアクセスしようとしたユーザは、認証を要求されません。 Web エージェントは、スラッシュが 3 つ登場した後で、それ以降の URI 部分を無視します。 たとえば、このパラメータを以下の値に設定したとします。

```
http://www.example.com/directory
```

Web エージェントは以下の URI を無視します。

```
directory
```

指定された URI が別のドメインにあっても、出現場所にかかわらず Web エージェントはそれを無視します。 たとえば、Web エージェントは、以下の URL のすべてに事前に表示された URI を無視します。

```
http://www.example.com/directory
http://www.example.net/directory
http://www.example.org/directory
```

注: この値では、大文字と小文字が区別されます。

デフォルト： デフォルトなし

例： （ローカル設定ファイル内の複数の URI）

IgnoreUrl="http://www.example.com/directory"

IgnoreUrl="http://www.example.com/directory2"

例： （ドメインを指定せずに URI のみを使用）

IgnoreUrl="/*resource*/"

URI への無制限のアクセスを許可するには、以下のタスクのいずれかを実行します。

- 中央設定では、無視する URI を持つ完全修飾ドメイン名をエージェント設定オブジェクトに追加します。URI が複数ある場合は、パラメータに複数値の設定を使用します。
- ローカル設定では、ローカル設定ファイルで完全修飾ドメイン名と URI ごとに個別の行を追加します。

指定された URI を使用するリソースは、Web エージェントによって無視され、それらのリソースへのアクセスが自動的に付与されます。

第 19 章: ログ記録およびトレース

このセクションには、以下のトピックが含まれています。

起動イベントのログ

デバッグを支援するために、起動時のイベントはログに記録されます。各メッセージは、問題の手がかりになる可能性があります。 これらのログは、以下の場所に格納されます。

- Windows システムでは、Windows アプリケーション イベント ログに記録されます。
- UNIX システムでは、STDERR へ送信されます。Apache サーバは、STDERR を Apache の error_log ファイルにマップするので、これらのイベントもそのログに記録されます。

エラー ログとトレース ログ

Web エージェントのパフォーマンスを監視したり、Web エージェントとポリシー サーバの通信状態を確認したりする場合は、Web エージェントのロギング機能を使用します。 ロギング機能は、パフォーマンスを分析して問題をトラブルシューティングすることを目的として、SiteMinder プロセスの動作に関する正確で包括的な情報を提供します。

ログとは、プログラムを実行している間に発生したイベントからなる記録のことです。1 つのログは、一連のログメッセージによって構成されています。各ログメッセージは、プログラムを実行している間に発生した何らかのイベントについて記述しています。 ログメッセージは、ログファイルに書き込まれます。

注: IIS 6.0 Web エージェントは、最初のユーザ要求が送信されてから、ログ ファイルを作成します。 Apache 2.0 Web エージェントでは Apache サーバの起動時にログ ファイルが作成されます。

Web エージェントは、以下のログ ファイルを使用します。

エラー ログ

プログラム レベルおよび操作レベルのエラーが含まれます。たとえば、エージェントがポリシー サーバと通信できない場合を示します。このログの詳細出力のレベルはカスタマイズできません。エラー ログに含まれるメッセージのタイプは、以下のとおりです。

エラーメッセージ

プログラム レベルのエラーが含まれます。これらは、プログラムの不適切または異常な動作、あるいはネットワーク障害のような何らかの外部の問題に起因すると考えられる機能障害を示しています。 動作レベルのエラーもあります。 このエラーのタイプは、ファイルを開く、またはユーザを認証するなどの動作の成功を妨げる障害を意味します。

情報メッセージ

何らかのイベントが発生したことをユーザまたは管理者に知らせることを目的とするメッセージです。つまり、サーバの起動や停止、または何らかのアクションが実施されたことを意味します。

警告メッセージ

通常とは異なる状況やイベント、または潜在的な問題を示唆している状況やイベントについて、ユーザまたは管理者に警告することを目的とするメッセージです。 これは必ずしも、何かが誤っていることを意味するとは限りません。

トレース ログ

詳細な警告メッセージと情報メッセージが含まれます。これらは、設定することができます。たとえば、トレース メッセージやフロー状態メッセージが含まれます。 また、このファイルには、ヘッダの詳細や cookie 変数などのデータも含まれます。トレース ログに含まれるメッセージは、以下のとおりです。

トレース メッセージ

トレースまたはデバッグ、あるいはその両方の目的で、プログラムの動作に関する詳細な情報を提供します。 トレース メッセージは一般的に、通常の動作時は無効にしておきます。 情報、警告、およびエラーの各メッセージとは対照的に、トレース メッセージはソース コード内に埋め込まれていて、容易にはローカライズできません。 さらに、トレース メッセージには、メッセージ自体の他に、重要なデータが含まれていることがあります。たとえば、現在のユーザやレルムの名前です。

Web エージェントを設定するときに、エラー ログ ファイルとトレース ログ ファイルの両方のロケーションを指定します。 エラー ログとトレース ログは、Web エージェントの正常な動作を妨げている可能性がある問題の解決に役立ちます。

注: Windows プラットフォーム上のエージェントで、EnableWebAgent パラメータを yes に設定すると、Web エージェント ログが確実に作成されます。EnableWebAgent を no（デフォルト）のままにして、ログ パラメータを設定した場合は、UNIX プラットフォーム上のエージェントに対するエージェント ログのみが作成されます。

詳細情報

エラー ロギングのセットアップと有効化 (P. 437)
トレース ロギングの設定 (P. 442)

ログ ファイルに表示されるパラメータ値

Web エージェントは、Web エージェントのエラー ログ ファイルに、設定パラメータとその値をリストしますが、トラディショナル エージェントとフレームワーク エージェントとでその方法は異なります。

フレームワーク エージェントでは、エージェント設定オブジェクトまたはローカル設定ファイルに入力されているとおりに、設定パラメータとその値がログ ファイルに正確に記録されます。 値が正しくない可能性があるパラメータを含め、すべてのパラメータがログ ファイルに記録されます。

トラディショナル エージェントでは、パラメータ値を記録する前に処理が行われます。 パラメータの値が正しい場合は、パラメータとその値がログ ファイルに記録されます。 値が正しくないパラメータは、ログ ファイルに記録 *されません*。

エラー ロギングのセットアップと有効化

エラー ログには次の設定が必要です。

- ロギングが有効になります。
- ログ ファイルの場所を指定する。

エラー ロギングを有効にするパラメータや、ログ データを追加するなどのオプションを指定するパラメータは、ローカル設定ファイル、またはポリシー サーバのエージェント設定オブジェクトに定義されます。

IIS または Apache の Web サーバにインストールされるエージェントは、ローカル設定ファイルでローカルに設定されるログ パラメータの動的な設定をサポートしません。 変更はエージェントが再起動されたときに有効になります。 ただし、これらのログ設定をポリシー サーバのエージェント設定オブジェクトに格納し、動的に更新することができます。

注: IIS 6.0 Web エージェントは、最初のユーザ要求が送信されてから、ログ ファイルを作成します。 Apache 2.0 Web エージェントでは Apache サーバの起動時にログ ファイルが作成されます。

次の手順に従ってください:

1. ログ ファイルがまだない場合は、ログ ファイルと関連ディレクトリを作成します。
2. LogFile パラメータの値を yes に設定します。

   注: Web サーバのローカル設定ファイル内でこのパラメータの値を Yes に設定すると、ポリシー サーバ上で定義されたあらゆるロギング設定を上書きします。 たとえば、このパラメータの値が LocalConfig.conf ファイルで yes に設定されていると仮定します。 たとえ対応するエージェント設定オブジェクトの AllowLocalConfig パラメータの値が no に設定されていたとしても、エージェントはログ ファイルを作成します。また、エージェント設定オブジェクト内の他の設定を上書きするために、LocalConfig.conf ファイル内の関連するロギング パラメータも設定できます。
3. ファイル名も含め、エラー ファイルの絶対パスを、以下のいずれかのパラメータで指定します。

   LogFileName

   ログ ファイルの完全パス（ファイル名を含む）を指定します。

   デフォルト： No

例： （Windows） *web_agent_home*¥log¥WebAgent.log

例： （UNIX/LInux）
/export/iPlanet/servers/https-jsmith/logs/WebAgent.log

LogFileName32

IIS （32 ビット アプリケーションを保護する 64 ビット Windows オペレーティング環境上にある）用の SiteMinder Web エージェントのログ ファイルの完全パスを指定します。 32 ビット アプリケーションは 64 ビット Windows オペレーティング環境で Wow64 モードで実行されます。 ロギングが有効であるが、このパラメータが設定されていない場合、IIS 用 Web エージェントはログ ファイル名の末尾に _32 を追加します。

デフォルト： No

制限： Windows 64 ビット オペレーティング環境のみ。 パスの最後にファイル名を指定します。

例： （Wow64 モードを使用する Windows 64 ビット オペレーティング環境） *web_agent_home*¥log¥WebAgent32.log

4. （省略可）以下のパラメータを設定します（ポリシー サーバのエージェント設定オブジェクト、またはローカル設定ファイルで設定します）。

LogAppend

既存のログ ファイルの最後に新しいログ情報を追加します。 このパラメータを no に設定した場合、ロギングが有効になるたびに、ログ ファイル全体が上書きされます。

デフォルト： No

LogFileSize

ログ ファイルのサイズ制限（メガバイト単位）を指定します。 現在のログ ファイルがこの制限に到達すると、新しいログ ファイルが作成されます。 新しいログ ファイルは、以下の命名規則のうちの 1 つを使用します。

- フレームワーク エージェントでは、新しいログ ファイルの名前は、元の名前にシーケンス番号が追加されたものになります。 たとえば、サイズ制限に到達すると、myfile.log という名前のログ ファイルは myfile.log.1 に名前が変更されます。

- 従来のエージェントでは、新しいログファイルの名前は、元の名前に日付とタイムスタンプが追加されたものになります。たとえば、サイズ制限に到達すると、myfile.log という名前のログファイルは myfile.log.09-18-2003-16-07-07 に名前が変更されます。

古いファイルは手動でアーカイブするか削除する必要があります。

デフォルト： 0（ロールオーバーなし）

例： 80

LogLocalTime

ログがグリニッジ標準時（GMT）とローカル時間のどちらを使用するかを指定します。GMT を使用するには、この設定を no に変更します。このパラメータが存在しない場合は、デフォルト設定が使用されます。

デフォルト： yes

ローカル設定ファイルを使用する場合、設定は以下の例のようになります。

```
LogFile="yes"
LogFileName="/export/iPlanet/servers/https-myserver/logs/errors.log"
LogAppend="no"
LogFileSize="80"
LogLocalTime="yes"
```

エラー ロギングが有効になります。

トランスポート層インターフェース（TLI）ロギングの有効化

エージェントとポリシー サーバの間の接続を検討する場合は、トランスポート層インターフェース ロギングを有効にします。

TLI ロギングを有効にする方法

1. Web サーバに以下の環境変数を追加します。

   SM_TLI_LOG_FILE

2. 以下の例に示されるように、変数の値のディレクトリおよびログ ファイル名を指定します。

   directory_name/log_file_name.log

3. エージェントが有効であることを確認します。
4. Web サーバを再起動します。

   TLI ロギングが有効になります。

保存されるログ ファイルの数の制限

エージェントが保持するログ ファイルの数を制限できます。 たとえば、エージェント ログを格納するシステム上のディスク空き容量を節約したい場合は、以下のパラメータを使用して、ログ ファイルの数を制限できます。

LogFilesToKeep

保持するエージェント ログ ファイルの数を指定します。以下の場合に新しいログ ファイルが作成されます。

- エージェントが起動したとき。
- ログ ファイルのサイズ制限（LogFileSize パラメータの値で指定）に達したとき。

このパラメータの値を変更しても、保持数を超える既存のログ ファイルは自動的に削除*されません*。たとえば、システムに 500 個のログ ファイルが格納されているときに、それらのファイルのうち 50 個のみを保持することを指定しても、エージェントは、残りの 450 個のファイルを削除*しません*。

このパラメータの値を 0 に設定すると、すべてのログ ファイルが保持されます。

デフォルト： 0

次の手順に従ってください:

1. 既存のログ ファイルをすべて、システムからアーカイブするか削除します。
2. LogAppend パラメータの値を no に設定します。
3. LogFilesToKeep パラメータの値を維持するログ ファイルの数に変更します。

トレース ロギングをセットアップする方法

トレース ロギングをセットアップするには、以下の手順に従います。

1. トレース ロギングをセットアップおよび有効化します。
2. 以下のリストを確認することによってトレース ログに記録する内容を決定します。
   - トレース ログ コンポーネントとサブコンポーネント
   - トレース メッセージ データ フィールド
   - データ フィールド フィルタ
3. デフォルトのトレース設定ファイルを複製します。
4. 記録するアイテムが含まれるように複製ファイルを変更します。
5. エージェントを再起動します。

トレース ロギングの設定

トレース ロギングを使用するには、事前に、トレース ログ ファイルの名前、ロケーション、およびパラメータを指定して、トレース ロギングを設定しておく必要があります。 これらの設定によって、ファイル自体のサイズおよび形式を制御します。トレース ロギングを設定したら、トレース ログ ファイルのコンテンツを別個に指定します。 これにより、トレース ログ ファイル自体のパラメータを変更することなく、必要に応じて、トレース ログに含める情報のタイプを変更することができます。

トレース ロギングを設定するには、以下の手順に従います。

1. Web サーバ上で WebAgentTrace.conf ファイルを見つけます。ファイルを複製します。

   注: IIS 用の SiteMinder エージェントを実行しており、64 ビット システム（WoW64 モード）上の 32 ビット アプリケーションを保護している場合は、2 つの複製を作成します。 64 ビット Windows 動作環境には、32 ビットと 64 ビットのアプリケーションに異なるディレクトリがあります。

2. エージェント設定オブジェクトまたはローカル設定ファイルを開きます。

3. TraceFile パラメータに yes を設定します。

   注: Web サーバのローカル設定ファイル内でこのパラメータの値を Yes に設定すると、ポリシー サーバ上で定義されたあらゆるロギング設定を上書きします。 たとえば、このパラメータの値が LocalConfig.conf ファイルで yes に設定されていると仮定します。 たとえ対応するエージェント設定オブジェクトの AllowLocalConfig パラメータの値が no に設定されていたとしても、エージェントはログ ファイルを作成します。また、エージェント設定オブジェクト内の他の設定を上書きするために、LocalConfig.conf ファイル内の関連するロギング パラメータも設定できます。

4. 以下のパラメータでトレース ログ ファイルの絶対パスを指定します。

   TraceFileName

   トレース ログ ファイルの完全パスを指定します。

   デフォルト： デフォルトなし
   例： C:¥Program Files¥ca¥webagent¥log¥trace.log

   TraceFileName32

64 ビット Windows オペレーティング環境で稼働し、32 ビット アプリケーションを保護している IIS 用 SiteMinder エージェントのトレース ファイルの完全パスを指定します。 64 ビット Windows オペレーティング環境にインストールされ、32 ビット Windows アプリケーションを保護している IIS 用 SiteMinder エージェントがある場合は、このパラメータを設定します。 32 ビット アプリケーションは 64 ビット Windows オペレーティング環境で Wow64 モードで実行されます。 トレース ロギングが有効であるが、このパラメータが設定されていない場合、IIS 用 Web エージェントはファイル名の末尾に _32 を追加します。

デフォルト： デフォルトなし

制限： Windows 64 ビット オペレーティング環境のみ。 パスの最後にトレース ファイル名を指定します。

例： （Wow64 モードを使用する Windows 64 ビット オペレーティング環境） web_agent_home¥log¥WebAgentTrace32.log

5. WebAgentTrace.conf ファイルのコピー（手順 1 で作成）のフル パスを以下のパラメータに指定します。

TraceConfigFile

監視するコンポーネントとイベントを決定する WebAgentTrace.conf 設定ファイルの場所を指定します。

デフォルト： デフォルトなし

例： C:¥Program Files¥ca¥webagent¥config¥WebAgentTrace.conf

TraceConfigFile32

監視するコンポーネントとイベントを決定する WebAgentTrace.conf 設定ファイルの場所を指定します。 64 ビット Windows 動作環境にインストールされ、32 ビット Windows アプリケーションを保護している IIS 用の SiteMinder エージェントがある場合は、このパラメータを設定します。 32 ビット アプリケーションは 64 ビット Windows 動作環境で Wow64 モードで実行されます。 ログ記録が有効であるが、このパラメータが設定されていない場合、IIS 用 Web エージェントはファイル名の末尾に _32 を追加します。

デフォルト： デフォルトなし

制限： Windows 64 ビット動作環境のみ。 パスの最後に設定ファイル名を指定します。

例： （Wow64 モードを使用する Windows 64 ビット動作環境）
web_agent_home¥config¥WebAgentTrace32.conf

注: このファイルは、Web サーバを再起動するまで使用されません。

6. エージェント設定オブジェクトまたはローカル設定ファイルに以下のパラメータを設定することで、トレース ログ ファイルの情報の形式を定義します。

TraceAppend

ロギングが有効になるたびにファイル全体を書き直す代わりに、既存のログ ファイルの最後に新しいログ情報を追加します。

デフォルト： No

TraceFormat

トレース ファイルがメッセージを表示する方法を指定します。 以下の*いずれか*のオプションを選択します。

- default - 角かっこ［］を使用してフィールドを囲みます。
- fixed - 固定幅のフィールドに使用します。
- delim - 選択した文字を使用してフィールドを区切ります。
- xml - XML-like タグを使用します。 Web エージェントには、DTD（Document Type Definition、文書タイプ定義）や、他のスタイルシートは*付属していません*。

デフォルト： default（角かっこ）

TraceDelimiter

トレース ファイル内のフィールドを区切るカスタム文字を指定します。

デフォルト： デフォルトなし

例： |

TraceFileSize

トレース ファイルの最大サイズを指定します（メガバイト単位）。この制限に到達すると、Web エージェントは新しいファイルを作成します。

デフォルト： 0（新しいログ ファイルは作成されません）

例： 20（MB）

LogLocalTime

ログがグリニッジ標準時（GMT）とローカル時間のどちらを使用するかを指定します。GMT を使用するには、この設定を no に変更します。このパラメータが存在しない場合は、デフォルト設定が使用されます。

デフォルト： yes

7. Web エージェントが目的のアクティビティを監視するように、WebAgentTrace.conf ファイルを編集します。

   フレームワーク Web エージェントは、エージェント設定ファイルでローカルに設定されたログ パラメータの動的な設定をサポートしていません。 したがって、パラメータを変更しても、Web サーバを再起動するまでは、その変更結果は有効になりません。 しかし、ポリシー サーバ上のエージェント設定オブジェクト内でそれらのパラメータを設定した場合、ログに関するそれらの設定は保存され、動的に更新されます。

   注: IIS 6.0 Web エージェントは、最初のユーザ要求が送信されてから、ログ ファイルを作成します。 Apache 2.0 Web エージェントでは Apache サーバの起動時にログ ファイルが作成されます。

8. Web エージェントで新しいトレース設定ファイルが使用されるよう、Web サーバを再起動します。

トレース ログ コンポーネントとサブコンポーネント

SiteMinder エージェントは、特定の SiteMinder コンポーネントを監視できます。 コンポーネントを監視すると、そのコンポーネントに対するすべてのイベントがトレース ログに記録されます。 各コンポーネントには 1 つ以上のサブコンポーネントがあり、エージェントはこれらも監視できます。 エージェントに、コンポーネントに対するイベントを必ずしもすべて記録させる必要がない場合は、監視する必要があるサブコンポーネントのみを指定することができます。

たとえば、Web サーバ上のエージェントに対するシングル サインオン メッセージのみを記録する場合は、WebAgent コンポーネントと SSO サブコンポーネントを指定します。

使用可能なコンポーネントとサブコンポーネントは、以下のとおりです。

AgentFramework

すべてのエージェント フレームワーク メッセージを記録します （フレームワーク エージェントにのみ適用されます）。使用可能なサブコンポーネントは、以下のとおりです。

- 管理
- フィルタ
- HighLevelAgent
- LowLevelAgent
- LowLevelAgentWP

AffiliateAgent

4.x のアフィリエイト エージェントに関連する Web エージェント メッセージを記録します。4.x のアフィリエイト エージェントは、Federation セキュリティ サービス (別途購入製品) の一部です（フレームワーク エージェントにのみ適用されます）。使用可能なサブコンポーネントは、以下のとおりです。

- RequestProcessing

SAMLAgent

SAML アフィリエイトエージェントに関連する Web エージェントメッセージです。 （フレームワーク エージェントにのみ適用されます）。使用可能なサブコンポーネントは、以下のとおりです。

- RequestProcessing

WebAgent

すべての Web エージェント ログ メッセージを記録します。 IIS 6.0 または Apache 2.0 エージェントを*除く*すべてのエージェントに適用されます。 使用可能なサブコンポーネントは、以下のとおりです。

- AgentCore
- キャッシュ
- 認証
- レスポンス
- 管理
- SSO
- フィルタ

Agent_Functions

すべてのエージェント API メッセージを記録します。 使用可能なサブコンポーネントは、以下のとおりです。

- Init
- UnInit
- IsProtected
- ログイン
- ChangePassword
- 確認
- ログアウト
- 認証
- 監査
- FreeAttributes
- UpdateAttributes
- GetSessionVariables
- SetSessionVariables
- DeleteSessionVariables
- トンネル
- GetConfig
- DoManagement

Agent_Con_Manager

エージェント API の内部処理に関連するメッセージを記録します。 使用可能なサブコンポーネントは、以下のとおりです。

- RequestHandler
- クラスタ
- サーバ
- WaitQueue
- 管理
- 統計

各サブコンポーネントの説明については、WebAgentTrace.conf ファイルを参照してください。

トレース メッセージ データ フィールド

メッセージに含めるデータ フィールドを指定することにより、特定のコンポーネントの各トレース メッセージに何を含めるかを定義することができます。

データ フィールドの構文は、以下のとおりです。

```
data:data_field1,data_field2,data_field3
```

以下の例に、いくつかのデータ フィールドを示します。

```
data:message,date,time,user,agentname,IPAddr
```

メッセージによっては、フィールドに対するデータが存在しないこともあるため、フィールドが空になる場合もあります。たとえば、データ フィールドとして RealmOID を選択した場合、トレース メッセージによって、レルムの OID が表示される場合と表示されない場合があります。

使用可能なデータ フィールドは、以下のとおりです。

Message

実際のトレース メッセージが含まれます。

SrcFile

トレース メッセージのソース ファイルと行番号が含まれます。

Pid

プロセス ID が含まれます。

Tid

スレッド ID が含まれます。

Date

日付が含まれます。

Time

時間が含まれます。

PreciseTime

ミリ秒単位までの時間が含まれます。

Function

トレース メッセージが入っているコード内の関数が含まれます。

User

ユーザの名前が含まれます。

Domain

SiteMinder ドメインが含まれます。

Realm

SiteMinder レルムが含まれます。

AgentName

使用されているエージェント名が含まれます。

TransactionID

トランザクション ID が含まれます。

DomainOID

SiteMinder ドメイン OID が含まれます。

IPAddr

クライアント IP アドレスが含まれます。

RequestIPAddr

エージェントがあるサーバの IP を表示するトレース ファイルが含まれます。

IPPort

クライアント IP ポートが含まれます。

CertSerial

証明書シリアル番号が含まれます。

SubjectDN

証明書の所有者 DN が含まれます。

IssuerDN

証明書の発行者 DN が含まれます。

SessionSpec

SiteMinder セッション仕様が含まれます。

SessionID

SiteMinder セッション ID が含まれます。

UserDN

ユーザ DN が含まれます。

Resource

要求されたリソースが含まれます。

Action

要求されたアクションが含まれます。

RealmOID

レルム OID が含まれます。

ResponseTime

CA Web エージェントまたは SDK エージェントと API アプリケーションに関連する、ポリシー サーバの平均レスポンス時間（ミリ秒単位）が含まれます。

注: ResponseTime をトレース ログに出力するには、WebAgentTrace.conf ファイル、またはポリシー サーバ設定オブジェクト（ACO）に指定されたその他のファイルに、データ フィールド ResponseTime と一緒にコンポーネント Agent_Con_Manager を含めて、ポリシー サーバを再起動してください。Agent_Con_Manager コンポーネント、つまりエージェント API 接続マネージャは、ポリシー サーバからレスポンスを受け取るたびに ResponseTime を計算して、実行の平均を維持します。 トレース ログで ResponseTime を見つけるには、［PrintStats］を検索してください。

トレース メッセージ データ フィールド フィルタ

特定の問題に焦点を当てるために、データ フィールドの値に基づいてフィルタを指定することによって、トレース ログの出力を絞り込むことができます。 たとえば、index.html ページで問題が発生している場合、トレース設定ファイル内で Resource:==/html と指定することにより、html サフィックス（拡張子）を持つリソースのみをフィルタして抽出することができます。 各フィルタは、ファイル内で個別の行を使用して記述する必要があります。

フィルタは、以下の構文を使用します。

data_field:*filter*

使用可能なフィルタのタイプは、以下のとおりです。

- ==（完全一致）
- !=（等しくない）

フィルタは、以下の例に示すように、ブール ロジックを使用します。

Action:!=get（get 以外のすべてのアクション）

Resource:==/html（末尾が /html のすべてのリソース）

トレース ログのコンテンツの決定

WebAgentTrace.conf ファイルによって、トレース ログのコンテンツが決まります。 トレース ログに表示するコンポーネントとデータ項目は、Web サーバ上の WebAgentTrace.conf ファイルの設定を変更することで制御できます。 ファイル編集時には、次の要素が当てはまります。

- エントリは、大文字と小文字が区別されます。

  コンポーネント、データ フィールド、またはフィルタを指定する場合、その値は、WebAgentTrace.conf ファイルの命令内にあるオプションと正確に一致している必要があります。

- 設定行のコメントを解除します。
- 既存のエージェントの上に新しいエージェントをインストールする前に WebAgentTrace.conf ファイルを変更すると、ファイルは上書きされます。 まず、ファイルの名前を変更するか、ファイルのバックアップを作成します。 インストール後、変更を新しいファイルに統合することができます。

次の手順に従ってください:

1. WebAgentTrace.conf ファイルを開きます。

   注: 元のファイルを複製し、コピーを変更することをお勧めします。コピーを変更することで、デフォルト設定が保存されます。

2. 以下のステップに従って、コンポーネントとサブコンポーネントを追加します。

   a. エージェントのタイプと一致するセクションを見つけます。 たとえば、サーバに Apache 2.0 エージェントがインストールされている場合は、以下の例のような行を探します。

      ```
      # For Apache 2.0, Apache 2.2, IIS 7.0 and SunOne Web Agents
      ```

      注: 詳細については、「SiteMinder Web エージェント インストール ガイド」を参照してください。

   b. そのセクションで以下の行を見つけます。

      ```
      #components:
      ```

   c. 行をコメント解除します。 次に、コロンの後に目的のコンポーネント名を追加します。 コンポーネントが複数ある場合は、以下の例のように、カンマで区切ります。

      ```
      components:  AgentFramework, HTTPAgent
      ```

   d. （任意）コンポーネント名の後ろに、目的のサブコンポーネントの名前を追加します。 以下の例のように、サブコンポーネント名はスラッシュで区切ります。

      ```
      components:  AgentFramework/Administration
      ```

3. 以下のステップに従って、データ フィールドとフィルタを追加します。
   a. 該当するセクションで以下の行を見つけます。

      ```
      #data:
      ```

   b. 行をコメント解除します。 次に、コロンの後に目的のデータ フィールドを追加します。 データ フィールドが複数ある場合は、以下の例のように、カンマで区切ります。

      ```
      data: Date, Time, Pid, Tid, TransactionID, Function, Message, IPAddr
      ```

   c. （オプション）データ フィールドの後ろに、コロン、ブール演算子、および目的の値を付けて、データ フィールドにフィルタを追加します。 フィルタに指定する値は、完全に一致する必要があります。 以下の例は、特定の IP アドレスのアクティビティをログに記録するフィルタを示しています。

      ```
      data: Date, Time, Pid, Tid, TransactionID, Function, Message,
      IPAddr:==127.0.0.1
      ```

      注: 各フィルタは、ファイル内の別々の行に指定する必要があります。
4. 変更を保存し、ファイルを閉じます。
5. 変更を適用するために Web サーバを再起動します。

   トレース ログのコンテンツが決定されました。

保存されるトレース ログ ファイルの数の制限

SiteMinder エージェントが保持するトレース ログの数を制限できます。たとえば、エージェント ログを格納するシステム上のディスク空き容量を節約したい場合は、以下のパラメータを使用して、トレース ログの数を制限できます。

TraceFilesToKeep

保持する SiteMinder エージェント トレース ログ ファイルの数を指定します。 以下の場合に新しいトレース ログが作成されます。

- エージェントが起動したとき。
- トレース ログのサイズ制限（TraceFileSize パラメータの値で指定）に達したとき。

このパラメータの値を変更しても、保持数を超える既存のトレース ログは自動的に削除*されません*。 たとえば、システムに 500 個のトレース ログが格納されているときに、それらのファイルのうち 50 個のみを保持することを指定しても、エージェントは、残りの 450 個のトレース ログを削除*しません*。

このパラメータの値を 0 に設定すると、すべてのトレース ログが保持されます。

デフォルト： 0

次の手順に従ってください:

1. 既存のトレース ログをすべて、システムからアーカイブするか削除します。
2. TraceAppend パラメータの値を no に設定します。
3. TraceFilesToKeep パラメータの値を維持するトレース ログの数に変更します。

Agent Connection Manager のトレース ログによる詳細なエージェント接続データの収集

Web エージェントとポリシー サーバの間の接続に関する詳細情報を収集するために、Agent Collection Manager によって収集された情報が含まれているトレース ログ ファイルを作成します。

詳細な Web エージェント接続データを収集する方法

1. エージェント設定オブジェクトまたはローカル設定ファイルを開きます。
2. TraceFile パラメータの値を yes に設定します。

   注: Web サーバのローカル設定ファイル内でこのパラメータの値を yes に設定すると、ポリシー サーバ上で定義されたあらゆるロギング設定より優先されます。 たとえば、LocalConfig.conf ファイル内でこのパラメータの値を yes に設定すると、ポリシー サーバ上の対応するエージェント設定オブジェクトで AllowLocalConfig パラメータの値を no に設定しても、ログ ファイルは生成されます。さらに、ポリシー サーバのトレース ログ設定をすべて上書きするには、LocalConfig.conf ファイル内の（ファイルの名前やサイズなどを定義する）関連するトレース ロギング パラメータを設定してください。
3. TraceFileName パラメータの中で、エージェント接続データのトレース ログ ファイルの絶対パスを指定します。 これは、トレース ログ出力を保持するファイルです。
4. 以下のファイルの絶対パスに TraceConfigFile パラメータの値を設定します。

   web_agent_home/config/AgentConMgr.conf

   web_agent_home

   SiteMinder エージェントがインストールされているディレクトリを示します。

   デフォルト（SiteMinder Web エージェントの Windows 32 ビット インストールのみ）： C:¥Program Files¥CA¥webagent

   デフォルト（Windows 64 ビット インストール［IIS 用 SiteMinder Web エージェントのみ］）： C:¥Program Files¥CA¥webagent¥win64

   デフォルト（64 ビット システムで稼働している Windows 32 ビット アプリケーション［IIS 用 SiteMinder Web エージェントを持つ Wow64 のみ］）： C:¥Program Files (x86)¥webagent¥win32

デフォルト（UNIX/Linux インストール）： /opt/ca/webagent

5. 以下のパラメータを設定することによって、エージェント接続データのトレース ログ ファイルの形式を指定します。

TraceAppend

ロギングが有効になるたびにファイル全体を書き直す代わりに、既存のログ ファイルの最後に新しいログ情報を追加します。

デフォルト： No

TraceDelimiter

トレース ファイル内のフィールドを区切るカスタム文字を指定します。

デフォルト： デフォルトなし

例： |

TraceFileSize

トレース ファイルの最大サイズを指定します（メガバイト単位）。この制限に到達すると、Web エージェントは新しいファイルを作成します。

デフォルト： 0（新しいログ ファイルは作成されません）

例： 20（MB）

TraceFormat

トレース ファイルがメッセージを表示する方法を指定します。 以下の*いずれかの*オプションを選択します。

- default - 角かっこ［］を使用してフィールドを囲みます。
- fixed - 固定幅のフィールドに使用します。
- delim - 選択した文字を使用してフィールドを区切ります。
- xml - XML-like タグを使用します。 Web エージェントには、DTD（Document Type Definition、文書タイプ定義）や、他のスタイルシートは*付属していません*。

デフォルト： default（角かっこ）

LogLocalTime

ログがグリニッジ標準時（GMT）とローカル時間のどちらを使用するかを指定します。GMT を使用するには、この設定を no に変更します。このパラメータが存在しない場合は、デフォルト設定が使用されます。

デフォルト： yes

6. Web サーバを再起動すると、新しい設定が有効になります。

   Web エージェント接続に関する詳細情報が収集されます。

   注: SiteMinder12.52 では、BusyHandleCount と FreeHandleCount の属性は使用されません。

第 20 章: トラブルシューティング

第 21 章: エージェント エラー コード

IIS 用エージェント トラブルシューティング ログ

問題の状況:

IIS 用 SiteMinder エージェントのトラブルシューティングに役立つ IIS 7.x ログ ファイルがあるか。

解決方法:

以下のログ ファイルを開きます。

web_agent_home¥log¥IIS70Trace.log

このログ ファイルには以下の種類の情報が含まれます。

- アプリケーション プール ID
- アプリケーション プール パイプライン モード
- フィルタ タイプ(ISAPI フィルタに対するネイティブ HTTP モジュール)
- 32 ビットまたは 64 ビット

ログ ファイルの重複した LLAWP エラー

問題の状況:

ログ ファイルに以下のエラーが表示されます。

```
重複した LLAWP
```

解決方法:

このエラーは、アプリケーション プールがリサイクルするときに発生します。 現在の LLAWP プロセスが完全にシャットダウンする前に、アプリケーション プールが新しい LLAWP プロセスを開始しようとしています。

このエラーを防ぐには、IIS Web サーバ内にアプリケーション プールを設定し、ローテーションのオーバーラップを禁止します。この設定により、アプリケーション プールは、現在の LLAWP プロセスが停止するのを待ってから、新しいプロセスを開始します。

注: 詳細については、IIS の Web サイトで「DisallowOverlappingRotation」を検索してください。

カスタム エラー ページが表示されない

Oracle Directory Enterprise Edition Oracle iPlanet Directory Server Enterprise Edition）で有効

問題の状況:

以下のいずれかの設定パラメータを設定しましたが、代わりにサーバから汎用的なエラー メッセージを受信しました。

CSSErrorFile

ユーザが可能なクロスサイト スクリプティング文字が含まれている URL を開こうとした場合に、ユーザに表示するカスタム エラー メッセージ ファイルまたは URL の場所を指定します。

デフォルト： デフォルトなし

ServerErrorFile

サーバ エラーに遭遇したユーザに対してカスタム エラー ページを表示するように Web エージェントに指示します。 このパラメータのファイル パスまたは URL を指定します。

デフォルト： デフォルトなし

解決方法:

以下の手順を実行します。

1. Web サーバ上の instance_name-obj.conf ファイルを開きます。
2. 次の行を検索します。

   ```
   AuthTrans fn="SiteMinderAgent"
   ```

3. 以下の例のように、この行の末尾に UseOutputStreamSize="0" を追加します。

   ```
   AuthTrans fn="SiteMinderAgent" UseOutputStreamSize="0"
   ```

4. ファイルを保存し、Web サーバを再起動します。

トレース メッセージを初期化できない

IIS 7.x で有効

問題の状況:

同じ IIS Web サーバで 32 ビットおよび 64 ビットのエージェントを実行しています。32 ビット エージェント用のトレース ログを記録したいのですが、以下のメッセージが代わりに表示されます。

［INFO］ LLAWP： トレースを初期化できません。

64 ビット エージェント トレース ログには影響しません。

解決方法:

以下の設定パラメータがエージェント用に定義されていることを確認します。

TraceConfigFile32

監視するコンポーネントとイベントを決定する WebAgentTrace.conf 設定ファイルの場所を指定します。 64 ビット Windows 動作環境にインストールされ、32 ビット Windows アプリケーションを保護している IIS 用の SiteMinder エージェントがある場合は、このパラメータを設定します。 32 ビット アプリケーションは 64 ビット Windows 動作環境で Wow64 モードで実行されます。ログ記録が有効であるが、このパラメータが設定されていない場合、IIS 用 Web エージェントはファイル名の末尾に _32 を追加します。

デフォルト： デフォルトなし

制限： Windows 64 ビット動作環境のみ。 パスの最後に設定ファイル名を指定します。

例： （Wow64 モードを使用する Windows 64 ビット動作環境）
web_agent_home¥config¥WebAgentTrace32.conf

LogFileName32

IIS（32 ビット アプリケーションを保護する 64 ビット Windows オペレーティング環境上にある）用の SiteMinder Web エージェントのログ ファイルの完全パスを指定します。 32 ビット アプリケーションは 64 ビット Windows オペレーティング環境で Wow64 モードで実行されます。 ロギングが有効であるが、このパラメータが設定されていない場合、IIS 用 Web エージェントはログ ファイル名の末尾に _32 を追加します。

デフォルト： No

制限： Windows 64 ビット オペレーティング環境のみ。 パスの最後にファイル名を指定します。

例： （Wow64 モードを使用する Windows 64 ビット オペレーティング環境） *web_agent_home*¥log¥WebAgent32.log

TraceFileName32

64 ビット Windows オペレーティング環境で稼働し、32 ビット アプリケーションを保護している IIS 用 SiteMinder エージェントのトレース ファイルの完全パスを指定します。 64 ビット Windows オペレーティング環境にインストールされ、32 ビット Windows アプリケーションを保護している IIS 用 SiteMinder エージェントがある場合は、このパラメータを設定します。 32 ビット アプリケーションは 64 ビット Windows オペレーティング環境で Wow64 モードで実行されます。 トレース ロギングが有効であるが、このパラメータが設定されていない場合、IIS 用 Web エージェントはファイル名の末尾に _32 を追加します。

デフォルト： デフォルトなし

制限： Windows 64 ビット オペレーティング環境のみ。 パスの最後にトレース ファイル名を指定します。

例： （Wow64 モードを使用する Windows 64 ビット オペレーティング環境） web_agent_home¥log¥WebAgentTrace32.log

エージェント サーバとポリシー サーバがファイアウォールで隔てられている場合に KeepAlives を有効にする

問題の状況:

エージェントとポリシー サーバの間にファイアウォールを使用します。ページにアクセスしようとすると、500 エラーがエージェントから返される場合があります。

解決方法:

以下の手順に従うことによりエージェント上の Keepalives を有効にします。

1. Web エージェントをホストしているコンピュータで、以下の環境変数を見つけます。

   ```
   SM_ENABLE_TCP_KEEPALIVE
   ```

2. 前の環境変数の値を 1 に設定します。

日本語ページが不適当に表示される（153202、153609）

問題の状況:

ユーザが以下のいずれかの理由での別のページにリダイレクトされるとき、結果ページのコンテンツは正しく表示されません。

- 新しい登録
- パスワードを変更する
- パスワード期限切れ

解決方法:

Apache Web サーバの httpd.conf ファイルに以下の行を追加します。

```
“BrowserMatch ".*" suppress-error-charset”
```

http.conf ファイルを保存し、この設定の Web サーバを再起動して有効にします。

英語以外の入力文字にジャンク文字が含まれる

問題の状況:

SiteMinder コンポーネントを UNIX マシン上にコンソール モードでインストールまたは設定する場合、大部分の英語以外の入力文字がコンソール ウィンドウに正しく表示されません。

解決方法:

コンソール ウィンドウの端末設定を確認し、以下のコマンドを実行して、コンソールが入力文字の高（第 8）ビットをクリアしないことを確認します。

```
stty –istrip
```

第 22 章: エージェント エラー コード

このセクションには、以下のトピックが含まれています。

00-0001

原因:

IP アドレスからエージェント名を特定できない

処置:

エージェント設定を参照し、Web サーバが提供する各 HOST アドレスに対応して AgentName がマップされていること、または DefaultAgentName が正しく設定されていることを確認します。

00-0002

原因:

問題のある文字が URL 内に存在するか、BadUrlChars パラメータ内で定義された文字が URL 内で検出されました。

処置:

以下のいずれかを実行します。

- 問題のある文字を URL から削除します
- その文字を BadUrlChars パラメータのリストから削除します。その結果、その URL はブロックされなくなります。

00-0004

原因:

SSLCRED cookie がエラーのステータスを示している。

処置:

SCC（安全な認証情報コレクタ）として動作している Web エージェントを調査し、その設定を確認します。

通常、このエラーが発生するのは、SCC エージェントが自らの環境から認証情報を取得できない場合のみです。 これは設定エラーの可能性を示しています。

00-0005

原因:

FORMCRED cookie がエラーのステータスを示している。

処置:

FCC（フォーム認証情報コレクタ）として動作している Web エージェントを調査し、その設定を確認します。

通常、このエラーが発生するのは、FCC エージェントが自らの環境から認証情報を取得できない場合のみです。 これは設定エラーの可能性を示しています。

00-0006

原因:

NTLM 保護されたリソースが、期待されているリソース キャッシュの中で見つからなかった。

処置:

Windows 認証方式のセットアップを調査し、設定を確認します。

00-0007

原因:

ASCII エンコード エラーが存在する。 これは Web エージェントの内部エラーです。

処置:

Web サーバと Web エージェントを調査し、不安定であることが疑われるサービスを診断します。

Web エージェントのログ ファイルと設定ファイルを参照できるように用意して、カスタマ サポートに問い合わせます。

詳細情報:

CA への連絡先 (P. 3)

00-0008

原因:

SSL 認証が失敗した。 このエラーは、不適切な証明書が存在すること、またはそのユーザが認証されていないことを示します。

処置:

他の証明書を使用するか、SSL 認証方式の設定を調査し、疑わしい原因を探します。

00-0009

原因:

SSL 認証情報が不適切または見つからない。

処置:

他の証明書、またはユーザ名とパスワードのペアを使用します。 SSL 認証方式の設定を調査し、疑わしい原因を探します。

00-0010

原因:

アクセスが拒否されました。 このエラーは、アクセスがブロックされたことに起因する、一般的な障害を示しています。

処置:

Web エージェントとポリシー サーバのログを調査し、障害の根本的な原因を判断します。

00-0011

原因:

認証情報コレクタのエラー。このエラーは、アクセスがブロックされたことに起因する、フォームベースまたは SSL ベースの高度な認証に関する一般的な障害を示しています。

処置:

以下の手順を実行します。

- Web エージェントとポリシー サーバのログを確認し、障害の根本的な原因を判断します。
- 高度な認証方式のセットアップを調査し、原因を調べます。

00-0012

原因:

暗号化エラー。これは Web エージェントの内部エラーを示唆しています。

処置:

以下の手順を実行します。

- Web サーバと Web エージェントを調査し、不安定であることが疑われるサービスを診断します。
- キー ストアのセットアップを参照し、適切なエージェント キーが使用されていることを確認します。
- カスタマ サポートに問い合わせ、Web エージェントのログ ファイルと設定ファイルを参照用に送ります。

詳細情報:

CA への連絡先 (P. 3)

00-0013

原因:

エージェント設定エラー。起動時に 1 つ以上のエラーが発生し、Web エージェントの有効な設定を行うことができませんでした。

処置:

以下の手順を実行します。

- Windows 環境では、［アプリケーション］イベント ログを参照し、詳細を把握します。
- Apache エージェントでは、Apache エラー ログを参照して、詳細を把握します。
- Oracle iPlanet UNIX エージェントでは、シェルのプロンプトから Oracle iPlanet を起動し、STDERR によって表示される中から、疑わしいエラーを探します。
- SmHost.conf ファイルが存在している（ホストが正しく登録された）こと、および正しいエントリが記録されていることを確認します。
- エージェント設定ファイル内に、有効な SmHost.conf ファイルを指す、1 つの有効な HostConfigFile エントリが存在していることを確認します。
- AgentConfigObject が、1 つの有効な値を保持していることを確認します。

00-0014

原因:

ユーザをログ アウトできなかった。

処置:

詳細については以下のファイルを確認してください。

- Web エージェント ログ ファイル
- Web エージェント トレース ファイル
- ポリシー サーバ ログ ファイル
- ポリシー サーバ トレース ファイル

00-0015

原因:

SiteMinder 監査サービスは監査要求に SM_AGENTAPI_NO で応答しました。

処置:

詳細については以下のファイルを確認してください。

- ポリシー サーバ ログ ファイル
- ポリシー サーバ トレース ファイル

00-0016

原因:

FQ ホスト名を解決できない。

処置:

Web エージェントのログを確認して、エージェントが解決しようとしているホスト名を特定します。 ホスト名が正しい場合は、エージェントが実行される Web サーバの DNS 設定を確認します。

00-0017

原因:

無効なリダイレクト ターゲットが見つかった。

処置:

このメッセージをレポートしている Web エージェントのログ ファイルを調べて、処理されている URL（通常は FCC または他の高度な認証 URL）を特定し、TARGET CGI パラメータの値が有効であると考えられるかどうかを判断します。

10-0001

原因:

'SERVER_NAME' HTTP 変数を読み込むことができない。

処置:

Web ブラウザおよび Web サーバが HTTP 1.0 に準拠していることを確認します。

10-0002

原因:

'URL' HTTP 変数を読み込むことができない。

処置:

Web ブラウザおよび Web サーバが HTTP 1.0 に準拠していることを確認します。

10-0003

原因:

'method' HTTP 変数を読みこむことができない。

処置:

Web ブラウザおよび Web サーバが HTTP 1.0 に準拠していることを確認します。

10-0004

原因:

'host' HTTP 変数を読み込むことができない。

処置:

Web ブラウザおよび Web サーバが HTTP 1.0 に準拠していることを確認します。

10-0005

原因:

'URI' HTTP 変数を読み込むことができない。

処置:

Web ブラウザおよび Web サーバが HTTP 1.0 に準拠していることを確認します。

10-0007

原因:

URL が長すぎる。

処置:

MaxUrlSize パラメータの設定を大きくします。デフォルトの設定値は 4,096 バイトです。

詳細情報:

URL の最大サイズの設定 (P. 344)

20-0001

原因:

SiteMinder 監査サーバに接続できないか、ポリシー サーバの予期できないエラーが発生した。

処置:

以下の手順を実行します。

- ポリシー サーバのログを参照し、エラーの詳細を調べます。
- ポリシー サーバに ping を行い、Web エージェントとポリシー サーバの接続状態を確認します。エージェントとポリシー サーバ間にファイアウォールが構築されている場合、以下のサービス ポートがブロックされていないことを確認します。
  - 44441（監査）
  - 44442（認証）
  - 44443（許可）

20-0002

原因:

SiteMinder 認証サーバに接続できないか、ポリシー サーバの予期できないエラーが発生した。

処置:

以下の手順を実行します。

- ポリシー サーバのログを参照し、エラーの詳細を調べます。
- ポリシー サーバに ping を行い、Web エージェントとポリシー サーバの接続状態を確認します。エージェントとポリシー サーバ間にファイアウォールが構築されている場合、以下のサービス ポートがブロックされていないことを確認します。
  - 44441（監査）
  - 44442（認証）
  - 44443（許可）

20-0003

原因:

SiteMinder 許可サーバに接続できないか、ポリシー サーバの予期できないエラーが発生した。

処置:

以下の手順を実行します。

- ポリシー サーバのログを参照し、エラーの詳細を調べます。
- ポリシー サーバに ping を行い、Web エージェントとポリシー サーバの接続状態を確認します。エージェントとポリシー サーバ間にファイアウォールが構築されている場合、以下のサービス ポートがブロックされていないことを確認します。
  - 44441（監査）
  - 44442（認証）
  - 44443（許可）

30-0026

原因:

パスワード サービスのリダイレクト URL が使用できない。

処置:

パスワード サービスのリダイレクト URL が設定されていることを確認します。

付録 A: エージェントのパラメータ

このセクションには、以下のトピックが含まれています。

エージェント設定パラメータの一覧

以下の表に、エージェント設定パラメータの一覧を示します。

設定対象パラメータ	参照手順
AcceptTPCookie	SDK サードパーティ Cookie のサポート (P. 115)
AgentConfigObject	ローカル設定ファイルのみにあるパラメータ (P. 41)
AgentName	AgentName と DefaultAgentName の値の設定 (P. 58)
AgentNamesAreFQHostNames	混在環境での認証情報コレクタの設定 (P. 231)
AgentWaitTime	ネットワーク遅延への対応 (P. 65)
AllowCacheHeaders	HTTP ヘッダ リソースのキャッシュ方法の制御 (P. 171)
AllowLocalConfig	ローカル設定の実装 (P. 42) ローカル設定パラメータの変更の制限 (P. 45)
AppendIISServerLog	IIS サーバ ログでのユーザ名およびトランザクション ID の記録 (P. 347)
autoauthorizeoptions	OPTIONS メソッドを使用するリソースへの自動アクセス許可 (P. 271)
BadCSSChars	クロスサイト スクリプティングからの Web サイトの保護 (P. 87)
BadFormChars	無効なフォーム文字の指定
BadQueryChars	無効なクエリ文字の指定 (P. 93)
BadUrlChars	無効な URL 文字の指定 (P. 95)
CacheAnonymous	匿名ユーザのキャッシング (P. 417)
CCCExt	Cookie プロバイダの指定 (P. 292)

設定対象パラメータ	参照手順
ConformToRFC2047	RFC 2047 への準拠の無効化 (P. 263)
ConstructFullPwsvcUrl	パスワード サービス リダイレクトでの完全修飾 URL の使用 (P. 263)
CookieDomain	シングル サインオンの設定方法 (P. 278)
CookieDomainScope	Cookie ドメイン解決の実装 (P. 113)
CookiePath	エージェント Cookie の Cookie パスの指定 (P. 110)
CookiePathScope	エージェント Cookie の Cookie パスの指定 (P. 110)
CookieProvider	シングル サインオンの設定方法 (P. 278)
CookieValidationPeriod	検証期間と期限切れになった Cookie URL での悪用からのセッション Cookie の保護 (P. 134)
CSSChecking	クロスサイト スクリプティングをチェックするための Web エージェントの設定 (P. 89)
CSSErrorFile	エラー処理をセットアップする方法 (P. 181)
Custom401ErrorFile	エラー処理をセットアップする方法 (P. 181)
CustomIpHeader	IP アドレス検証の設定 (P. 169)
DecodeQueryData	クエリ データのデコード (P. 117)
DefaultAgentName	AgentName と DefaultAgentName の値の設定 (P. 58)
DefaultHostName	HOST ヘッダを送信しないテスティング ツールへの対応 (P. 408)
DefaultPassword	IIS プロキシ ユーザ アカウントの使用 (P. 363)
DefaultUserName	IIS プロキシ ユーザ アカウントの使用 (P. 363)
DeleteCerts	Stronghold サーバからの証明書の削除 (P. 374)
DisableAuthSrcVars	デフォルトの HTTP ヘッダ変数の無効化 (P. 177)
DisableDirectoryList	Sun Java System サーバ上でのディレクトリ参照の制限 (P. 375)
DisableDNSLookups	DNS DOS 攻撃の防止 (P. 98)
DisableDotDotRule	複雑な URI の処理 (P. 119)
DisableSessionVars	デフォルトの HTTP ヘッダ変数の無効化 (P. 177)
DisableUserNameVars	デフォルトの HTTP ヘッダ変数の無効化 (P. 177)

設定対象パラメータ	参照手順
DisableWindowsSecurityContext	IIS 用のエージェント上の Windows セキュリティ コンテキストの無効化 (P. 365)。
DisallowUTF8NonCanonical	クロスサイト スクリプティング攻撃からの J2EE アプリケーションの保護 (P. 90)
DLPExclusionList	DLP コンテンツ分類からのリソースの除外 **注**: 詳細については、「SiteMinder 実装ガイド」を参照してください。
DLPSupportEnabled	SharePoint エージェント設定オブジェクトの変更 **注**: 詳細については、「SiteMinder 実装ガイド」を参照してください。
DominoDefaultUser	Domino サーバによるユーザ認証 (P. 391)
DominoLegacyDocumentSupport	Lotus Notes ドキュメントに対してユーザがリクエストしたアクションの処理 (P. 402)
DominoLookUpHeaderForLogin	認証用の SiteMinder ヘッダの使用 (P. 397)
DominoMapUrlForRedirect	Domino Web エージェントによる FCC リダイレクト用 URL のマップ (P. 204)
DominoNormalizeUrls	Domino Web エージェントによる FCC リダイレクト用 URL のマップ (P. 204)
DominoSuperUser	Domino スーパー ユーザとしての認証 (P. 392)
DominoUseHeaderForLogin	認証用の SiteMinder ヘッダの使用 (P. 397)
DominoUserForAnonAuth	Domino での匿名 SiteMinder 認証方式の使用 (P. 398)
EarlyCookieCommit	IIS 用のエージェントの Cookie 設定タイミングの決定 (P. 367)
EnableAuditing	ユーザ アクティビティを追跡するように監査を設定 (P. 84)
EnableCookieProvider	Cookie プロバイダの無効化 (P. 293)
EnableFCCWindowsAuth	Windows 認証を許可するように FCC を設定 (P. 215)
EnableFormCache	フォーム キャッシュの設定 (P. 229)
EnableIntroscopeApiSupport	CA Technologies Wily Introscope を使用した Web エージェントの監視 (P. 422)
EnableMonitoring	OneView モニタによる Web エージェントの監視 (P. 421)
EnableOtherAuthTrans	複数の AuthTrans 機能の処理 (P. 376)

設定対象パラメータ	参照手順
EnableWebAgent	Web エージェントの有効化 (P. 77)
EncryptAgentName	エージェント名の暗号化 (P. 64)
EnforceRealmTimeouts	複数レルム間でタイムアウトを適用する方法 (P. 142)
ExpiredCookieURL	検証期間と期限切れになった Cookie URL での悪用からのセッション Cookie の保護 (P. 134)
ExpireForProxy	プロキシ サーバの背後にあるエージェントの設定 (P. 233)
FCCCompatMode	混在環境での FCC と NTC の使用 (P. 233)
FCCExt	IIS と Domino の各 Web サーバでの認証情報コレクタのセットアップ
FCCForceIsProtected	FCC によるフォーム認証用のレルム コンテキストの確立の強制 (P. 227)
Fcchtmlencoding	Web エージェント FCC ページのクロスサイト スクリプティング攻撃の防止
ForceCookieDomain	Cookie ドメインの強制 (P. 112)
ForceFQHost	Cookie ドメインの強制 (P. 112)
ForceIISProxyUser	IIS プロキシ ユーザ アカウントの使用 (P. 363)
FormCacheTimeout	フォーム キャッシュの設定 (P. 229)
GetPortFromHeaders	ポート番号に関する HTTP HOST リクエストの使用 (P. 370)
HostConfigFile	ローカル設定ファイルのみにあるパラメータ (P. 41)
HTTPHeaderEncodingSpec	HTTP ヘッダのエンコード仕様の設定
HttpsPorts	HTTPS ポートの定義 (P. 116)
IdleTimeoutURL	セッション タイムアウト後のユーザのリダイレクト (P. 140)
IgnoreCPForNotprotected	保護されていないリソースにおける Cookie プロバイダの無視 (P. 289)
IgnoreExt	保護されていないリソースのファイル拡張子を無視することによるオーバーヘッドの削減 (P. 424)
IgnoreHost	Web エージェントによって無視される仮想サーバの指定 (P. 190)
IgnoreQueryData	クエリ データの無視 (P. 428)

設定対象パラメータ	参照手順
IgnoreUrl	URI への無制限のアクセスの許可 (P. 430)
IISCacheDisable	Cookie が含まれるサーバ レスポンスのキャッシュの防止 (P. 366)
LegacyCookieProvider	FCC レルム コンテキスト確認の無効化によるパフォーマンスの向上 (P. 227)
LegacyEncoding	レガシー URL エンコードの受け入れ (P. 407)
LegacyStreamingBehavior	POST リクエストでのコンテンツ タイプの転送方法の選択 (P. 372)
LegacyTransferEncodingBehavior	Apache Web エージェントでのレガシー アプリケーションの使用 (P. 370)
LegacyVariables	HTTP ヘッダのレガシー変数の有効化
LimitCookieProvider	Cookie プロバイダ機能の制限 (P. 280)
LoadPlugin	フレームワーク エージェントの WebAgent.conf ファイル (P. 37)
localconfigfile	フレームワーク エージェントの WebAgent.conf ファイル (P. 37)
LogAppend	エラー ロギングのセットアップと有効化 (P. 437)
LogFile	エラー ロギングのセットアップと有効化 (P. 437)
LogFileName	エラー ロギングのセットアップと有効化 (P. 437)
LogFileSize	エラー ロギングのセットアップと有効化 (P. 437)
LogFilesToKeep	保存されるログ ファイルの数の制限 (P. 440)
LogLocalTime	エラー ロギングのセットアップと有効化 (P. 437)
LogoffUri	シングル サインオンでの完全ログオフの設定方法 (P. 298)
LowerCaseHTTP	ヘッダ内の小文字 HTTP の使用 (P. 175)
LowerCaseProtocolSpecifier	小文字の URL プロトコルの指定 (P. 209)
MasterCookiePath	エージェント Cookie の Cookie パスの指定 (P. 110)
MaxResourceCacheSize	リソース キャッシュの最大サイズの設定 (P. 418)
MaxSessionCacheSize	ユーザ セッション キャッシュの最大サイズの設定 (P. 419)
MaxTimeoutURL	セッション タイムアウト後のユーザのリダイレクト (P. 140)

設定対象パラメータ	参照手順
MaxUrlSize	URL の最大サイズの設定 (P. 344)
NTCExt	NTLM 認証情報コレクタの指定 (P. 229)
OverlookSessionAsPattern	セッション Cookie の作成または更新の防止
OverlookSessionForMethods	セッション Cookie の作成または更新の防止
OverlookSessionForMethodUri	メソッドと URI に基づいたセッション Cookie の作成または更新の防止 (P. 137)
OverlookSessionForUrls	セッション Cookie の作成または更新の防止
OverrideIgnoreExtFilter	拡張子のないリソースの保護 (P. 99)
P3PCompactPolicy	Web エージェントの設定による P3P コンパクト ポリシーへの対応 (P. 122)
PersistentCookies	永続的 Cookie の設定 (P. 109)
PersistentIPCheck	セキュリティ侵害を防止するための IP アドレスの比較 (P. 104)
PostPreservationFile	フレームワーク エージェントと従来のエージェントの間の POST 保存の有効化 (P. 205)
PreserveHeaders	HTTP ヘッダの保存 (P. 170)
PreservePostData	POST 保存の有効化または無効化 (P. 100)
ProxyAgent	SiteMinder リバース プロキシ展開の考慮事項 (P. 335)
ProxyDefinition	SiteMinder リバース プロキシ展開の考慮事項 (P. 335)
ProxyHeadersAutoAuth	Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ (P. 318)
ProxyHeadersAutoAuth10	Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ (P. 318)
ProxyHeadersProtected	Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ (P. 318)
ProxyHeadersProtected10	Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ (P. 318)
ProxyHeadersUnprotected	Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ (P. 318)

設定対象パラメータ	参照手順
ProxyHeadersUnprotected10	Cache-Control ヘッダ設定と ExpireForProxy ヘッダ設定のカスタマイズ (P. 318)
ProxyTimeout	SiteMinder リバース プロキシ展開の考慮事項 (P. 335)
ProxyTrust	プロキシ サーバの背後にあるエージェントの設定 (P. 316)
PSPollInterval	エージェントがポリシーまたはキーの更新をチェックする頻度の変更 (P. 82)
RemoteUserVar	REMOTE_USER 変数を設定するように Web エージェントを設定する (P. 154)
ReqCookieErrorFile	エラー処理のセットアップ (P. 181)
RequireClientIP	IP アドレス検証の設定 (P. 169)
RequireCookies	基本認証用の Cookie が必要です。 (P. 106)
ResourceCacheTimeout	リソース エントリをキャッシュに保存しておく時間の制御 (P. 420)
SaveCredsTimeout	保存された認証情報のタイムアウトの設定 (P. 415)
SCCExt	IIS と Domino の各 Web サーバでの認証情報コレクタのセットアップ
SecureApps	アプリケーションのセキュリティ保護 (P. 101)
SecureURLs	シングル サインオンと併用する場合の SecureUrls の設定 (P. 291)
ServerErrorFile	エラー処理のセットアップ (P. 181)
SessionGracePeriod	セッション猶予期間の変更 (P. 132)
SessionUpdatePeriod	セッション更新期間の変更 (P. 133)
SetRemoteUser	REMOTE_USER 変数を設定するように Web エージェントを設定する (P. 154)
SFCCExt	IIS と Domino の各 Web サーバでの認証情報コレクタのセットアップ
SkipDominoAuth	Domino サーバによるユーザ認証 (P. 391)
SSOTrustedZone	信頼とフェールオーバの順序 (P. 314)
SSOZoneName	セキュリティ ゾーンの設定 (P. 310)
StoreSessioninServer	使い捨てのセッション Cookie の有効化 (P. 138)

設定対象パラメータ	参照手順
SuppressServerHeader	URLScan ユーティリティを使用する場合のサーバ HTTP ヘッダの削除 (P. 342)
TargetAsRelativeURI	認証情報コレクタのリダイレクトでの相対ターゲットの使用 (P. 201)
TraceAppend	トレース ロギングの設定 (P. 442)
TraceConfigFile	トレース ロギングの設定 (P. 442)
TraceDelimiter	トレース ロギングの設定 (P. 442)
TraceFile	トレース ロギングの設定 (P. 442)
TraceFileName	トレース ロギングの設定 (P. 442)
TraceFileSize	トレース ロギングの設定 (P. 442)
TraceFilesToKeep	保存されるトレース ログ ファイルの数の制限 (P. 455)
TraceFormat	トレース ロギングの設定 (P. 442)
TrackCPSessionDomain	Cookie プロバイダ リプレイ攻撃の防御 (P. 281)
TrackSessionDomain	セッション Cookie ドメインの検証 (P. 139)
TransientIDCookies	識別 Cookie の制御 (P. 108)
TransientIPCheck	セキュリティ侵害を防止するための IP アドレスの比較 (P. 104)
UseAnonAccess	匿名ユーザ アクセスの有効化 (P. 364)
UseDominoUserForUnprotected	保護されていないリソースを認証するように Domino サーバで強制 (P. 406)
UseHTTPOnlyCookies	HTTP 専用属性を備えた Cookie での情報の保護 (P. 107)
UseNetBIOSforIISAuth	IIS 認証での NetBIOS 名または UPN の使用 (P. 349)
UseSecureCookies	安全な Cookie の設定 (P. 107)
UseSecureCPCookies	複数ドメインにわたる安全な Cookie の設定 (P. 288)
UseServerRequestIp	IP アドレスによるエージェント ID の解決 (P. 103)
ValidFedTargetDomain	有効なフェデレーション ターゲット ドメインの定義 (P. 202)
ValidTargetDomain	有効なターゲット ドメインの定義 (P. 89)
WebAppClientResponse	Web アプリケーション クライアントへの SiteMinder 動作の適用 (P. 123)

設定対象パラメータ	参照手順
XFrameOptions	カスタム レスポンスの X-Frame Options への準拠の確認 (P. 102)

第 23 章: SiteMinder サポート マトリックス

プラットフォーム サポート マトリックスへのアクセス

プラットフォーム サポート マトリックスを使用して、オペレーティング環境および他の必要なサードパーティ コンポーネントがサポートされていることを確認します。

次の手順に従ってください:

1. CA サポート サイトにログインします。
2. ［テクニカル サポート］セクションに移動します。
3. ［Product Finder］フィールドに「SiteMinder」と入力します。

   SiteMinder 製品ページが表示されます。
4. ［Product Status］-［SiteMinder Family of Products Platform Support Matrices］をクリックします。

注: 最新の JDK および JRE バージョンは、Oracle Developer Network でダウンロードできます。