目次

改訂情報

変更年月日	変更内容
2013-10-01	初版
2014-04-01	第2版 下記を追加・変更しました ■ 「*システム概要* 」に Webサービス 実行時のテナントについて追記 ■ 「*<wsTenantIdResolveType>タグ* 」を追加 ■ 「*wsse:TenantNotResolved* 」を追加 ■ 「*wsse:InvalidLoginGroupID* 」を追加 ■ 「*wsse:TenantIdNotMatch* 」を追加 ■ 「*wsse:FailedTenantSwitch* 」を追加
2016-04-01	第3版 下記を追加・変更しました ■ 「*WSSE認証モジュール*」にパスワードの保存方式が「ハッシュ化」の場合に利用できない旨の警告を追加
2016-08-01	第4版 下記を追加・変更しました ■ 「*認証・認可*」にTERASOLUNA Server Framework for Java (5.x) プログラミングガイドへのリンクを追記しました。
2017-04-01	第5版 下記を追加・変更しました ■ 「*付録*」の「Webサービス用ツール」のダウンロード方法を修正しました。
2020-12-01	第6版 下記を追加・変更しました ■ 「*付録*」の「*Apache Axis2 管理コンソールについて*」に注意書きを追記 ■ 「*付録*」の「*Apache Axis2 管理コンソールで Axis2 モジュール 「im_ws_auth」を指定する*」に注意書きを追記

はじめに

本書の目的

本書では intra-mart Accel Platform における Webサービス およびその認証・認可機能について説明します。

説明範囲は以下のとおりです。

- Webサービスの概要
- Webサービスの認証・認可

対象読者

本書では次の利用者を対象としています。

- intra-mart Accel Platform の Webサービス を管理する運用担当者
- Webサービス・プロバイダ の提供を行うアプリケーションの開発者
- Webサービス・クライアント を利用したアプリケーションの開発者

注意事項

1. Webサービス を利用するにあたり、いくつかの制限事項が存在します。制限事項についての詳細は「リリースノート」-「Webサービスの制限事項」を参照してください。

本書の構成

- *概要*

  Webサービス の概要について説明します。

- *認証・認可*

  intra-mart Accel Platform 上にデプロイされた Webサービス の認証・認可について説明します。

- *認証・認可のSOAPフォルト・コード*

  Webサービス 実行時に 認証・認可機能にて発生する SOAPフォルト・コード について説明します。

- *付録*

  Webサービス の管理や開発を行う上でに有益な情報を紹介します。

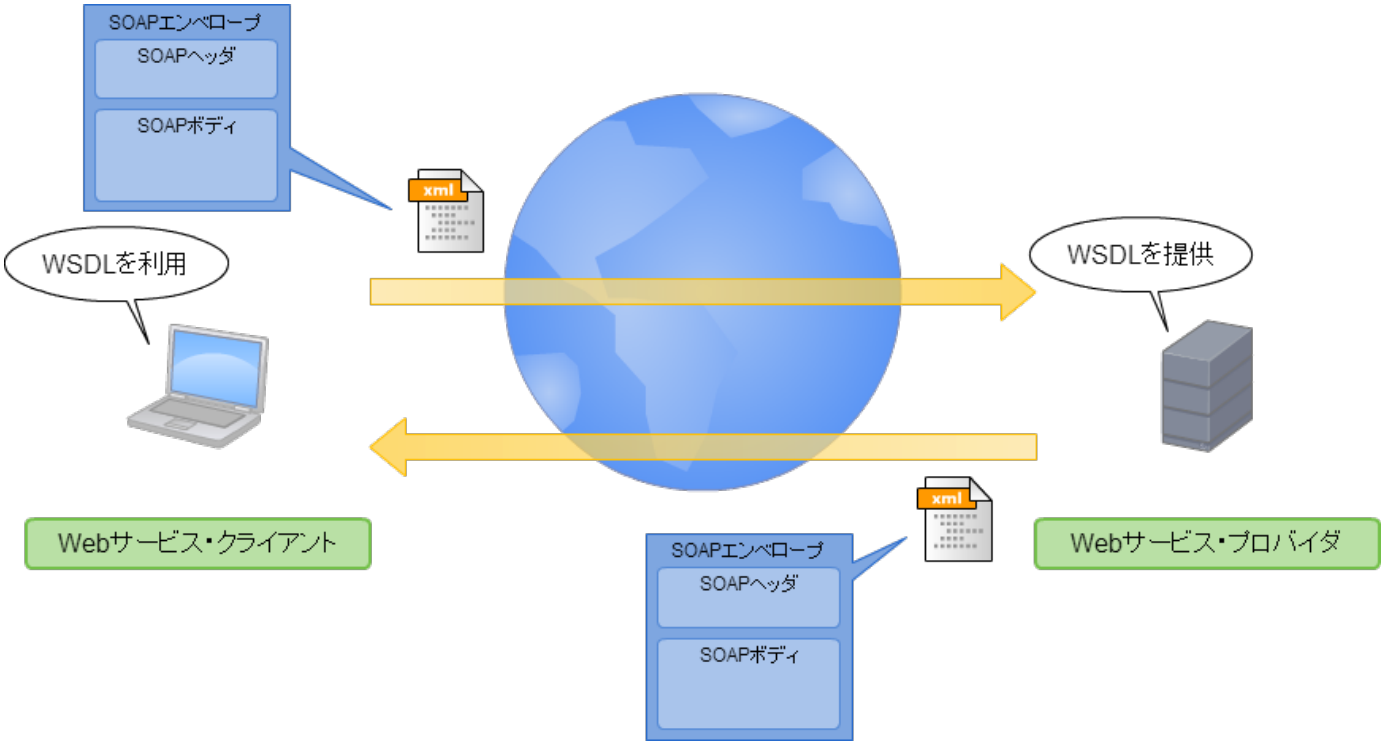
概要

Webサービスとは

「Webサービス」は、インターネットを用いた技術やそれらを利用したサービスを指しますが、本書で扱う「Webサービス」とは SOAP と WSDL を用いた Webサービス を指します。

Webサービス・プロバイダ と Webサービス・クライアント

本書では、Webサービス を提供する側を「Webサービス・プロバイダ」と呼び、Webサービス を利用する側を「Webサービス・クライアント」と呼びます。



SOAP

SOAP とはXML形式のメッセージを交換するためのプロトコルです。
Webサービス は、SOAP のプロトコルに従いXMLメッセージ（SOAPエンベロープ）を交換します。

SOAPエンベロープ は以下の２つで構成されます。

- SOAPヘッダ
  付加的な情報が格納されます。オプショナルです。
- SOAPボディ
  主要な情報（データ）が格納されます。

SOAP は、ソフトウェア同士がデータを交換するためのプロトコルです。SOAPエンベロープ には、通信プロトコルに関する記述が存在しないため、様々な場面で共通して利用することが可能です。

SOAP に関する詳細な情報は、書籍やWebサイトを参照してください。なお、SOAP の仕様はW3Cの「Web Services Activity」を参照してください。

SOAPフォルト

SOAP では、Webサービス・プロバイダ で何らかのエラーが発生した場合、SOAPボディ にエラー情報を格納して Webサービス・クライアント に返信することができます。このエラー情報を「SOAPフォルト」と呼びます。
SOAPフォルト は、エラーコード、エラーメッセージおよびエラーの詳細で構成されます。

WSDL

WSDL(Web Services Description Language) とは、Webサービス を記述するためのXMLをベースとした言語仕様です。
Webサービス を利用する際に必要となる以下の情報が記述されています。

- Webサービス のアクセスポイント
- Webサービス の通信プロトコル
- Webサービス を利用するための入力情報
- Webサービス の実行結果の形式

Webサービス・クライアント は WSDL に記述された Webサービス のインタフェースを元に、Webサービス を利用します。
WSDL は以下の要素で構成されています。

- `wsdl:definitions`
- `wsdl:types`
- `wsdl:message`
- `wsdl:operation`
- `wsdl:postType`
- `wsdl:binding`
- `wsdl:port`
- `wsdl:service`

WSDL に関する詳細な情報は、書籍やWebサイトを参照してください。なお、WSDL の仕様はW3Cの「Web Services Activity」を参照してください。

XMLスキーマ

WSDL の `wsdl:types` 要素は、Webサービス でやり取りされるXMLメッセージのフォーマットを定義しています。XMLメッセージの構造を定義するスキーマ言語として「XML Schema」を利用します。

XMLスキーマ に関する詳細な情報は、書籍やWebサイトを参照してください。なお、XMLスキーマ の仕様はW3Cの「XML Schema」を参照してください。

Webサービス・オペレーション

Webサービス・オペレーション とは、Webサービス で定義されている操作を意味します。Webサービス・オペレーション は、WSDL の `wsdl:operation` 要素で定義されています。

Apache Axis2

intra-mart Accel Platform では Webサービス エンジンとして「Apache Axis2」を採用しています。

認証・認可

機能概要

intra-mart Accel Platform では、Webサービス 実行時に、認証、認可および intra-mart Accel Platform へのログインを行います。
認証・認可を用いて Webサービス・クライアント から利用可能な Webサービス を制限します。

認証

受信したユーザ情報のユーザが intra-mart Accel Platform を利用可能であるかを確認します。
認証に失敗した場合、Webサービス を実行することはできません。
これにより、ユーザベースでのアクセス制御を行うことが可能です。

認可

受信したユーザ情報のユーザによる Webサービス・オペレーション の実行が「許可」されているかどうかを確認します。
これにより、権限ベースでのアクセス制御を行うことが可能です。

認可に関する詳細な情報は、「認可仕様書」を参照してください。

Webサービス への認可リソースの設定はservices.xmlで行います。
詳細は「*services.xmlに認可リソースを指定する*」を参照してください。

管理者による認可設定方法については、「*Webサービス・オペレーション への認可設定*」を参照してください。

ログイン

認証、認可が行われた後、ログインを行います。これにより Webサービス 実装者はログイン処理を意識することなく、業務処理の実装を行うことが可能です。
加えて、既存の業務処理の再利用が容易になります。具体的には アクセスコンテキスト を利用した処理を Webサービス の業務処理としてそのまま利用可能になります。

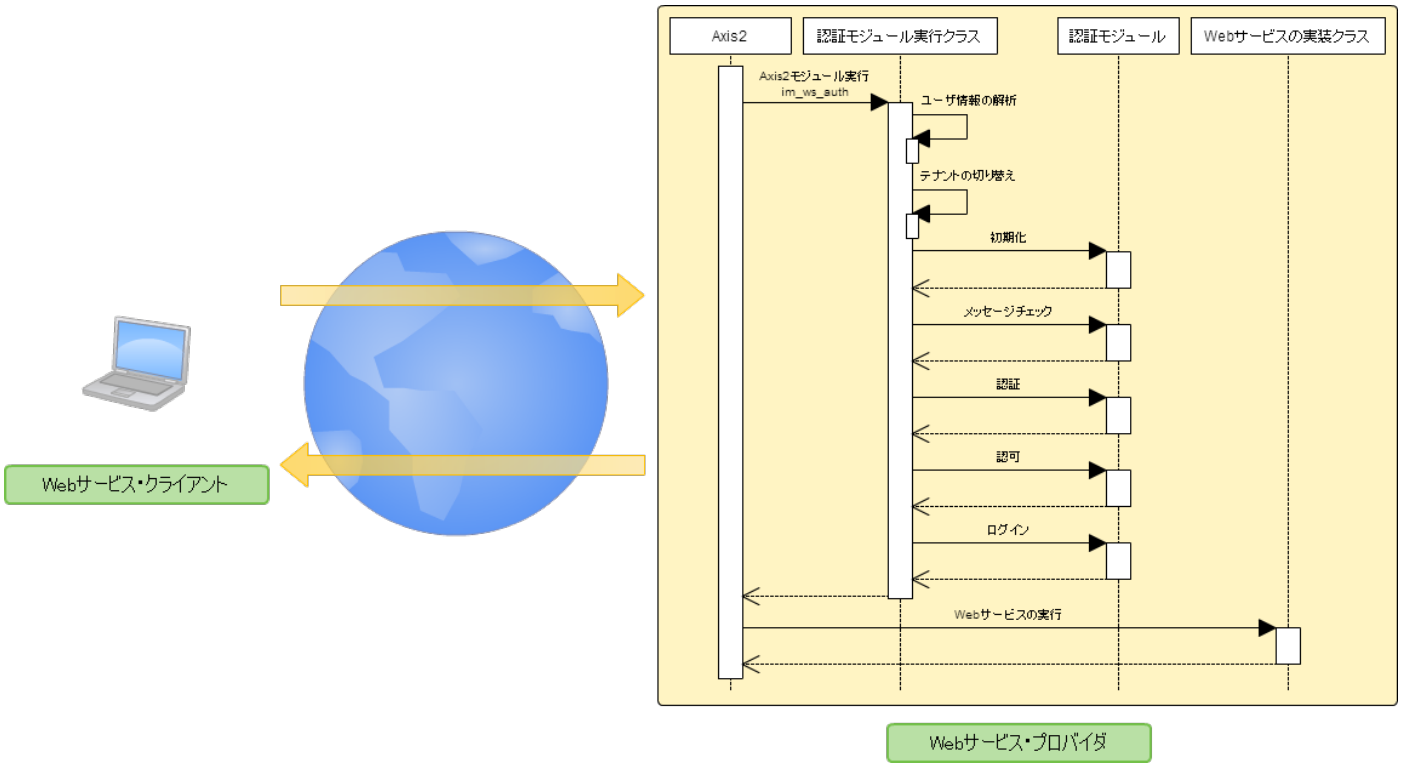
アクセスコンテキスト に関する詳細な情報は以下のドキュメントを参照してください。

- 「 SAStruts+S2JDBC プログラミングガイド 」-「 アクセスコンテキスト 」
- 「 スクリプト開発モデル プログラミングガイド 」-「 アクセスコンテキスト 」
- 「 TERASOLUNA Server Framework for Java (5.x) プログラミングガイド 」-「 アクセスコンテキスト 」

システム概要

認証・認可の流れ

Webサービス に対する認証・認可の流れは以下の通りです。



SOAPボディ に格納されたユーザ情報を元に認証・認可を行います。詳細は「*認証モジュール*」を参照してください。

ユーザ情報

ユーザ情報は、以下で構成されています。

- ログイングループID
- ユーザID
- 認証タイプ
- パスワード

ログイングループIDに指定する値については、*ログイングループID* を参照してください。
ユーザIDにはアカウントのユーザコードを指定します。
認証タイプは Webサービス・プロバイダ で利用する *認証モジュール* を指定します。
パスワードに格納する内容は、認証タイプによって異なります。

ログイングループID

認証・認可と Webサービス を実行する対象のテナントを指定します。

指定するべき値は、intra-mart Accel Platform のバージョンや設定により異なります。

intra-mart Accel Platform 2013 Winter 以前の場合

ログイングループIDには特に何も指定しなくてもかまいません。（指定した値により 認証・認可や Webサービス の動作が変わることはありません。）

intra-mart Accel Platform 2014 Spring 以降の場合

ログイングループIDには Webサービス 実行対象となるテナントのテナントIDを指定します。
ログイングループIDを省略した場合の動作は以下の通りです。

リクエスト情報を利用したテナント自動解決機能を利用している場合

リクエスト情報を利用して自動解決されたテナントで認証・認可と Webサービス の実行を行います。
ただし、*Webサービス・プロバイダ の認証・認可の設定* の *wsTenantIdResolveType* 設定が strict である場合は省略できません。
（省略した場合、SOAPフォルト *wsse:InvalidLoginGroupID* が発生します。）

リクエスト情報を利用したテナント自動解決機能を利用していない場合

デフォルトテナントで認証・認可と Webサービス の実行を行います。

> **コラム**
> 「リクエスト情報を利用したテナント自動解決機能」については、セットアップガイドの「テナント解決機能」を参照してください。

WSDL におけるユーザ情報の定義について

ユーザ情報 で述べたユーザ情報を受け渡すためのメッセージ形式を WSDL に定義する必要があります。
具体的には、下記のXMLスキーマで定義される型 WSUserInfo を Webサービス・オペレーション の第 1 引数に要素名 wsUserInfo として定義する必要があります。

```
<xs:schema
  attributeFormDefault="qualified"
  elementFormDefault="qualified"
  targetNamespace="http://auth.web_service.foundation.intra_mart.co.jp/xsd">
  <xs:complexType name="WSUserInfo">
    <xs:sequence>
      <xs:element minOccurs="0" name="authType" nillable="true" type="xs:string"/>
      <xs:element minOccurs="0" name="loginGroupID" nillable="true" type="xs:string"/>
      <xs:element minOccurs="0" name="password" nillable="true" type="xs:string"/>
      <xs:element minOccurs="0" name="userID" nillable="true" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

> **コラム**
> 引数の要素名 wsUserInfo は *Webサービス・プロバイダ の認証・認可の設定* の *<wsUserInfoArgumentName>タグ* で設定しています。

> **コラム**
> 認証・認可を必要としない Webサービス についてはこの限りではありません。
> 詳細は「*認証・認可を必要としない Webサービス について*」を参照してください。

認証モジュール実行クラス

認証モジュール を実行するための org.apache.axis2.engine.Handler を実装した Apache Axis2 ハンドラです。
このハンドラは Webサービス・クライアント が送信したユーザ情報を解析後、認証タイプで特定される *認証モジュール* に認証・認可処理を委譲しています。

intra-mart Accel Platform 2014 Spring 以降では、ユーザ情報の *ログイングループID* の値により、認証・認可と Webサービス を実行する対象となるテナントへの切り替えを行います。

このクラスは Axis2 モジュール 「im_ws_auth」が適用されている Webサービス に対して実行されます。

認証モジュール

認証・認可・ログインを行います。
詳細は、後述の「*認証モジュール*」を参照してください。

Webサービス の実装クラス（Webサービス・プロバイダ）

Webサービス として公開する業務処理が記述されたクラスです。

認証モジュール の各処理がすべて正常終了した後に、Webサービス・クライアント が要求した Webサービス・オペレーション が実行されます。そのため、Webサービス の実装クラスは認証・認可・ログインを考慮することなく業務処理を記述することが可能です。

Webサービス 実行後はログアウトを行います。従って、Webサービス 実行中のみユーザ情報で指定したユーザがログイン状態となります。

Webサービス・オペレーション の認証・認可・ログインを行うには、Webサービス・オペレーション に対して以下の２つの条件が満たされている必要があります。

- Axis2 モジュール 「im_ws_auth」が適用されていること
- *ユーザ情報* を格納するための引数が第１引数に付与されていること

Webサービス・クライアント

Webサービス・クライアント では、WSDL からスタブを作成するなどして、Webサービス を呼び出します。WSDL の中にはユーザ情報を受け渡すためのメッセージ形式が定義されています。（*WSDL におけるユーザ情報の定義について*）
これに則り、Webサービス・クライアント はユーザ情報を設定します。

Webサービス・プロバイダ で実行される *認証モジュール* は、Webサービス・クライアント で指定された *ユーザ情報* の認証タイプにより決定します。
認証タイプが未設定の場合、平文パスワード用認証モジュール が利用されます。
平文パスワード用認証モジュール についての詳細は、「*平文パスワード用認証モジュール*」を参照してください。

> **注意**
>
> Webサービス・プロバイダ に存在しない認証タイプを指定した場合は、SOAPフォルト （*wsse:BadRequest*）が発生します。

認証モジュール

認証モジュールでは、認証、認可およびログインを行います。
Webサービス 実行時に使用する　認証モジュールは Webサービス・クライアント から送信されたユーザ情報の認証タイプにより、決定します。
認証、認可およびログインで行う処理は利用する認証モジュールごとに異なります。各認証モジュールでの、認証・認可およびログイン処理については後述の各認証モジュールの仕様を参照してください。
認証モジュールは `jp.co.intra_mart.foundation.web_service.auth.WSAuthModule` インタフェースを利用して動作しています。
認証モジュールは、以下の順番で処理を行います。

1. 初期化（`WSAuthModule#init(WSUserInfo, MessageContext)` が実行されます。）
2. SOAP メッセージのチェック（`WSAuthModule#check(WSUserInfo, MessageContext)` が実行されます。）
3. 認証（`WSAuthModule#authentication(WSUserInfo)` が実行されます。）
4. 認可（`WSAuthModule#authorization(WSUserInfo, String, String)` が実行されます。）
5. ログイン（`WSAuthModule#login(WSUserInfo)` が実行されます。）

この一連の処理は、Webサービス・クライアント から要求があるたびに実行されます。
認証・認可などすべての処理に成功した場合のみ指定された Webサービス・オペレーション が実行されます。

各処理のいずれかに失敗するとエラーコードを格納した SOAPフォルト が Webサービス・クライアント に返却されます。
エラーコードの詳細については「*認証・認可のSOAPフォルト・コード*」を参照してください。

WSAuthModuleの詳細は、「WSAuthModuleインタフェースのAPIリスト」を参照してください。

intra-mart Accel Platform では標準で以下の認証モジュールが提供されています。

認証タイプ	提供モジュール	実装クラス名	概要
WSSE	Webサービス 認証・認可	WSAuthModule4WSSE	パスワードのダイジェスト化方法に、WS-SecurityのUsernameToken形式を採用した認証モジュール。
PlainTextPassword	Webサービス 認証・認可	WSAuthModule4PlainTextPassword	平文パスワード用認証モジュール。
Anonymous	Webサービス 認証・認可	WSAuthModule4Anonymous	未認証ユーザ用認証モジュール。

WSSE認証モジュール

WSSE認証モジュール は、パスワード・ダイジェスト 化方式にWS-SeurityのUsernameToken形式を採用した認証モジュールです。
完全修飾クラス名は `jp.co.intra_mart.foundation.web_service.auth.impl.WSAuthModule4WSSE` です。

Webサービス・クライアント は、以下を元に パスワード・ダイジェスト を作成します。

- Webサービス・クライアント が作成した「Nonce」
- Webサービス・クライアント が作成した「Created」
- Webサービス・クライアント が管理している認証対象ユーザの「パスワード」

WS-SecurityのUsernameToken形式の認証用文字列（以降、WSSE認証文字列）の具体例を示します。

```
UsernameToken Username="the_who", PasswordDigest="tLDSsdGqfvraHRh8BpqTYRBVy+U=",
Nonce="YTBiMWI2OGI2OTE3N2RlZQ==", Created="1966-12-01T12:34:56Z"
```

Webサービス・プロバイダ の認証モジュールでは、WSSE認証文字列 を解析し以下を元に パスワード・ダイジェスト を作成します。

- Webサービス・クライアント から送られてきた「Nonce」
- Webサービス・クライアント から送られてきた「Created」
- Webサービス・プロバイダ のサーバで管理されている認証対象ユーザの「パスワード」

Webサービス・クライアント から送信された パスワード・ダイジェスト と Webサービス・プロバイダ で作成した パスワード・ダイジェスト を比較し、その結果が同一であれば該当ユーザである（認証に成功）と判断します。同一でない場合は不正なユーザであると判別します。

WSSE認証文字列 の各項目は以下の通りです。

項目	説明
Username	ユーザ名（ユーザコード）。
Nonce	ランダムな値をBase64エンコードした文字列。
Created	Nonceが作成された日時をISO-8601表記で記述した文字列。
PasswordDigest	Nonce、Createdおよびパスワードを文字列連結し、SHA-1アルゴリズムでダイジェスト化して生成された文字列を、Base64エンコーディングした文字列。 具体的な作成方法は以下の通りです。 PasswordDigest = Base64 （ SHA-1 （ Nonce + Created + パスワード））

認可はユーザ情報で指定されたユーザIDのユーザに対して行います。

コラム

WSSE認証モジュール は パスワード・ダイジェスト の有効期限チェックを行います。
Createdの時間と比較し有効期限が経過している場合、認証に失敗します。有効期限の初期値は5分です。

有効期限の設定については「*WSSE認証モジュール <expire> タグ*」を参照してください。

コラム

WSSE認証文字列 の詳細は、「Webサービスセキュリティ・ユーザネームトークン・プロファイル 1.0 - 3 UsernameToken Extensions」を参照してください。

上記の仕様と WSSE認証モジュール で生成する パスワード・ダイジェスト の相違点は以下の通りです。

- NonceおよびCreatedが必須である
- Nonceの符号化の種類が常にBase64である

注意

受信した WSSE認証文字列 のCreatedとNonceを有効期限まで Webサービス・プロバイダ で保持します。既に保持されているCreatedとNonceで認証しようとした場合、認証に失敗します。このチェックは、クラスタ単位で行います。

注意

認証対象のユーザは *ユーザ情報* から特定します。WSSE認証文字列 のUsernameは利用しません。

注意

パスワードの保存方式が「ハッシュ化」方式である場合に利用できません。
パスワードの保存方式については「 システム管理者操作ガイド 」-「 パスワード保存方式設定 」を参照してください。

WSSE パスワード・ダイジェスト 生成API

認証タイプWSSEに対応した パスワード・ダイジェスト 生成用のユーティリティAPIは以下の通りです。

- スクリプト開発モデル API

  `WSAuthDigestGenerator4WSSE` オブジェクト

- Java API

  `jp.co.intra_mart.foundation.web_service.util.impl.WSAuthDigestGenerator4WSSE`

このAPIは Webサービス 認証・認可クライアント モジュールに同梱されています。
詳細な情報は、「WSAuthDigestGenerator4WSSEオブジェクトのAPIリスト」、または「WSAuthDigestGenerator4WSSEクラスのAPIリスト」を参照してください。

平文パスワード用認証モジュール

平文パスワード用認証モジュール は、パスワードを平文で送受信するための認証モジュールです。
完全修飾クラス名は `jp.co.intra_mart.foundation.web_service.auth.impl.WSAuthModule4PlainTextPassword` です。

ユーザ情報で指定されたパスワードが Webサービス・プロバイダ のサーバに登録されているアカウントのパスワードと一致した場合該当ユーザである（認証に成功）と判断します。
認可はユーザ情報で指定されたユーザIDのユーザに対して行います。

コラム

Webサービス・クライアント から送信される認証タイプが未設定の場合、この認証モジュールを利用します。

> **注意**
>
> 平文パスワード用認証モジュール は標準では利用できません。
> *Webサービス・プロバイダ の認証・認可の設定* の *<enablePlainTextPassword>タグ* を true に設定することで利用できます。

> **注意**
>
> 平文パスワード用認証モジュール はパスワードが平文で送信されるため、SOAP メッセージの中身やネットワーク通信内容を閲覧可能な利用者による成り済ましが可能になることに注意してください。
> この認証モジュールはSSLのような外部のセキュアなシステムと併用されないのであれば利用すべきではありません。

未認証ユーザ用認証モジュール

未認証ユーザ用認証モジュール は、未認証ユーザ で Webサービス を実行するための認証モジュールです。
完全修飾クラス名は jp.co.intra_mart.foundation.web_service.auth.impl.WSAuthModule4Anonymous です。
未認証ユーザ とは、ログインを行っていないユーザのことを指します。（ゲストユーザともいいます）

この認証タイプを利用した場合、認証処理は行われません。
認可処理は特定のユーザに対してではなく、未認証ユーザ に対して認可を行います。

ユーザ情報に格納されている、ユーザIDとパスワードは無視されます。

> **注意**
>
> 未認証ユーザ用認証モジュール は標準では利用できません。設定方法については「*認証モジュールを追加する*」を参照してください。

独自の認証モジュール利用する

以下を行うことで独自に認証モジュールを利用することも可能です。

- jp.co.intra_mart.foundation.web_service.auth.WSAuthModule インタフェースを実装したクラスを作成する
- %CONTEXT_PATH%/WEB-INF/conf/axis2.xml に <authModule> タグを設定する

WSAuthModuleの詳細は、「WSAuthModuleインタフェースのAPIリスト」を参照してください。

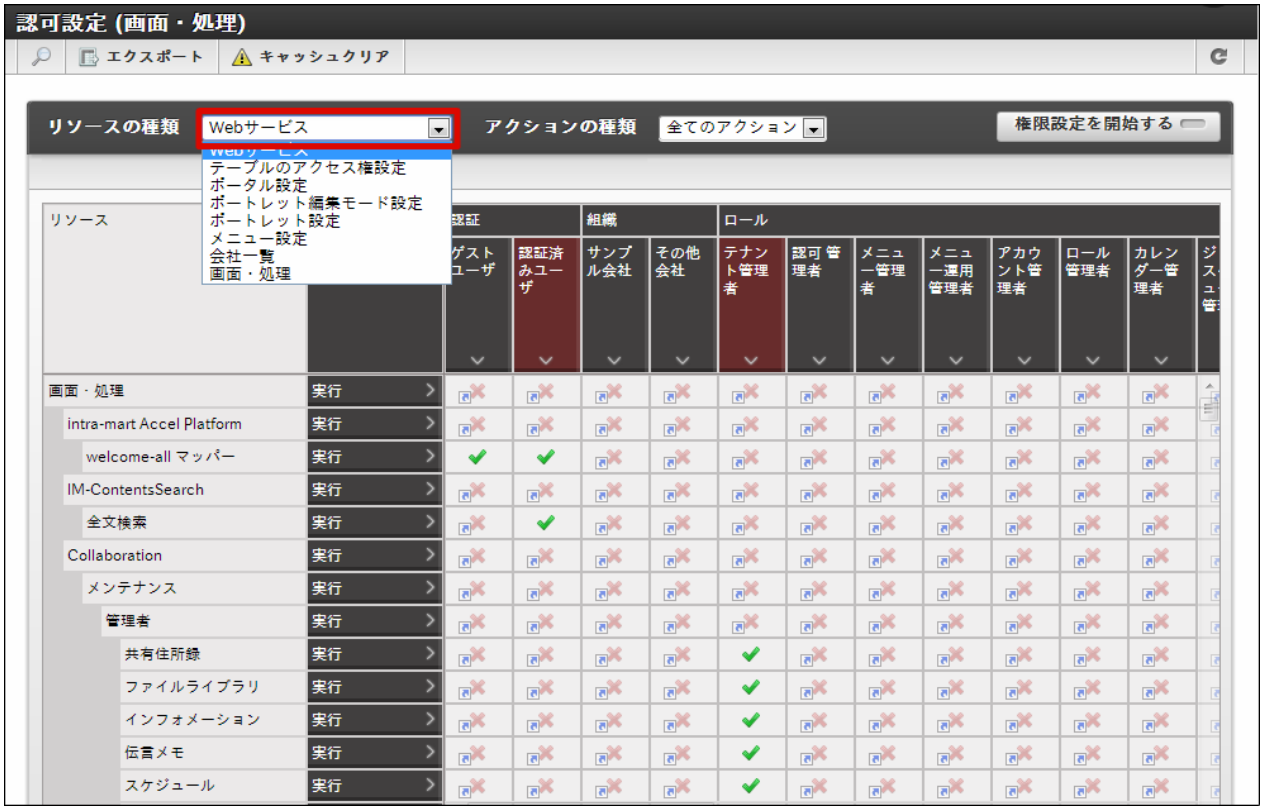
%CONTEXT_PATH%/WEB-INF/conf/axis2.xml の詳細は「*Webサービス・プロバイダ の認証・認可の設定*」を参照してください。

作成した認証モジュールの設定方法については「*認証モジュールを追加する*」を参照してください。

Webサービス・オペレーション への認可設定

intra-mart Accel Platform では認可設定により Webサービス・オペレーション へのアクセス権限を設定します。

1. 「サイトマップ」→「テナント管理」→「認可」をクリックします。
2. リソースの種類を「Webサービス」に変更します。



3. 「権限設定を開始する」をクリックし、認可設定を行います。

認可設定画面の詳細な操作方法は、「テナント管理者操作ガイド 認可を設定する」を参照してください。

> **コラム**
>
> 認可設定画面を利用するには、標準では以下のいずれかを満たすユーザでログインする必要があります。
>
> - テナント管理者ロールを保持している
> - 認可管理者ロールを保持している

Webサービス・プロバイダ の認証・認可の設定

Webサービス・プロバイダ による認証・認可の設定は %CONTEXT_PATH%/WEB-INF/conf/axis2.xml で行います。
設定は、parameter名 `jp.co.intra_mart.foundation.web_service` の子要素として記述します。

> **コラム**
>
> IM-Juggling では設定ファイル出力機能にて「Webサービス 認証・認可」の「Axis2設定(axis2.xml)」より出力が行えます。

> **注意**
>
> この設定は Axis2 モジュール 「im_ws_auth」 が適用されている Webサービス すべてに対して影響します。

以下は設定例です。

```
<axisconfig name="AxisJava2.0">

  <!-- ================================================= -->
  <!-- Parameters for intra-mart -->
  <!-- ================================================= -->
  <parameter name="jp.co.intra_mart.foundation.web_service">

    <enablePlainTextPassword>false</enablePlainTextPassword>

    <authModule class="jp.co.intra_mart.foundation.web_service.auth.impl.WSAuthModule4WSSE">
      <expire>300</expire>
    </authModule>

    <enableAuthentication>true</enableAuthentication>
    <enableAuthorization>true</enableAuthorization>

    <showSoapFaultDetail>true</showSoapFaultDetail>
    <wsUserInfoArgumentName>wsUserInfo</wsUserInfoArgumentName>
    <wsTenantIdResolveType>standard</wsTenantIdResolveType>

  </parameter>

  <!-- 省略 -->

</axisconfig>
```

<authModule>タグ

```
<authModule class="jp.co.intra_mart.foundation.web_service.util.impl.WSAuthDigestGenerator4WSSE">
  <expire>300</expire>
</authModule>
```

必須項目	×
複数設定	○
設定値・設定する内容	なし
単位・型	なし
省略時のデフォルト値	なし
親タグ	parameter

【属性】

属性名	説明	必須	デフォルト値
class	認証モジュールを設定します。`WSAuthModule` インタフェースを実装したクラスの完全修飾クラス名を指定します。	○	なし

> **コラム**
> `authModule` の子要素は設定値として扱われます。設定値は `WSAuthModule#setConfiguration(OMElement)` メソッドの引数に渡されます。

<enablePlainTextPassword>タグ

```
<enablePlainTextPassword>false</enablePlainTextPassword>
```

必須項目	×

複数設定	×
設定値・設定する内容	false　平文パスワード用認証モジュール を利用しません。 true　平文パスワード用認証モジュール を利用することを許可します。
単位・型	真偽値 (true/false)
省略時のデフォルト値	false
親タグ	parameter

【属性】

なし

<enableAuthentication>タグ

```
<enableAuthentication>true</enableAuthentication>
```

必須項目	×
複数設定	×
設定値・設定する内容	true　Webサービス 実行前に *認証* を実行します。 false　Webサービス 実行前に *認証* を実行しません。
単位・型	真偽値 (true/false)
省略時のデフォルト値	true
親タグ	parameter

【属性】

なし

<enableAuthorization>タグ

```
<enableAuthorization>true</enableAuthorization>
```

必須項目	×
複数設定	×
設定値・設定する内容	true　Webサービス 実行前に *認可* を実行します。 false　Webサービス 実行前に *認可* を実行しません。
単位・型	真偽値 (true/false)
省略時のデフォルト値	true
親タグ	parameter

【属性】

なし

<showSoapFaultDetail>タグ

```
<showSoapFaultDetail>true</showSoapFaultDetail>
```

必須項目	×
複数設定	×
設定値・設定する内容	true　認証モジュールで行われる処理に失敗した際に *SOAPフォルト* に詳細メッセージを含めます。 false　認証モジュールで行われる処理に失敗した際に *SOAPフォルト* に詳細メッセージを含めません。
単位・型	真偽値 (true/false)
省略時のデフォルト値	true
親タグ	parameter

【属性】

なし

> **注意**
> この設定で行う SOAPフォルト に詳細メッセージを含めるか否かは、認証モジュールで発生した SOAPフォルト についてのみ有効です。Webサービス で発生した SOAPフォルト には影響しません。

<wsUserInfoArgumentName>タグ

```
<wsUserInfoArgumentName>wsUserInfo</wsUserInfoArgumentName>
```

必須項目	×
複数設定	×
設定値・設定する内容	ユーザ情報を格納するための引数名。
単位・型	文字列
省略時のデフォルト値	wsUserInfo
親タグ	parameter

【属性】

なし

<wsTenantIdResolveType>タグ

この設定は intra-mart Accel Platform 2014 Spring 以降、利用可能です。

```
<wsTenantIdResolveType>standard</wsTenantIdResolveType>
```

必須項目	×
複数設定	×

設定値・設定する内容	ユーザ情報に含まれる *ログイングループID* の検証モードを指定します。指定可能な値と動作については、以下の通りです。 standard　とくに検証を行いません。 strict　ユーザ情報に含まれる loginGroupID と自動解決されたテナントIDが一致する場合のみ、認証を許可します。 ユーザ情報に含まれる loginGroupID と自動解決されたテナントIDが一致しない場合は、SOAPフォルト *wsse:TenantIdNotMatch* が発生します。 アクセス先のテナントに対して、予期しないテナントIDによるWebサービスの実行を防ぐ場合、この設定を使用します。 この設定は、「リクエスト情報を利用したテナント自動解決機能」を利用している場合のみ、利用してください。
単位・型	文字列
省略時のデフォルト値	standard
親タグ	parameter

【属性】

なし

> **コラム**
> 「リクエスト情報を利用したテナント自動解決機能」については、セットアップガイドの「テナント解決機能」を参照してください。

認証モジュール固有設定

intra-mart Accel Platform が標準で提供する認証モジュールで利用可能な設定について記述します。

WSSE認証モジュール <expire>タグ

```
<authModule class="jp.co.intra_mart.foundation.web_service.util.impl.WSAuthDigestGenerator4WSSE">
  <expire>300</expire>
</authModule>
```

必須項目	×
複数設定	×
設定値・設定する内容	認証時に利用されるユーザ情報の有効期限。 システム日時と WSSE認証文字列 のCreated を比較し、ここで設定された期限が過ぎている場合、認証に失敗します。 WSSE認証文字列 のCreatedとNonceをここで設定された期間、サーバ側で保持されます。既に保持されているCreatedとNonceで認証を行った場合、認証に失敗します。（リプレイ攻撃対策） この設定が「0」の場合、有効期限チェックおよびCreatedとNonceの保持は行いません。
単位・型	数値型（秒）
省略時のデフォルト値	300
親タグ	（class 属性が jp.co.intra_mart.foundation.web_service.util.impl.WSAuthDigestGenerator4WSSE である）authModule

【属性】

なし

項目

- 認証・認可のSOAPフォルト・コード
  - wsse:InvaildRequest
  - wsse:BadRequest
  - wsse:AuthenticationBadElements
  - wsse:ExpiredData
  - wsse:InvalidSecurityToken
  - wsse:FailedAuthentication
    - アカウントが登録されていません。
    - アカウントライセンスが登録されていません。
    - アカウントの有効期限が切れています。
    - アカウントがロックされています。
    - パスワードが間違っています。
  - wsse:RequestFailed
  - wsse:TenantNotResolved
  - wsse:InvalidLoginGroupID
  - wsse:TenantIdNotMatch
  - wsse:FailedTenantSwitch

認証・認可のSOAPフォルト・コード

認証・認可およびログイン時にエラーが発生した場合、そのエラー内容に対応する SOAPフォルト・コード が Webサービス・クライアント に返却されます。
以下に SOAPフォルト・コード とその原因を記述します。

以下に記述されていない SOAPフォルト・コード が送信された場合、業務処理内で何らかのエラーが発生している可能性があります。
業務処理で SOAPフォルト・コード が明示的に指定されていない場合は SOAPフォルト・コード は名前空間 `http://www.w3.org/2003/05/soap-envelope` で定義されている `Receiver` として送信されます。

> **コラム**
> *Webサービス・プロバイダ の認証・認可の設定* の *<showSoapFaultDetail>タグ* が `false` に設定されている場合、Webサービス・クライアント にはエラー詳細情報が通知されません。

> **注意**
> エラーメッセージの内容はロケールごとに異なるため注意してください。
> SOAP Fault Code や SOAP Fault Reasonのコード（[E.IWP.WEBSERVICE.AXIS2.XXXXX]など）はロケールに影響しません。

wsse:InvaildRequest

SOAP Fault Code	wsse:InvalidRequest
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00005] 要求が無効か、形式が間違っています。
SOAP Fault Detail（発生時の例）	java.lang.IllegalStateException: [E.IWP.WEBSERVICE.AXIS2.00007] wsUserInfo が見つかりませんでした。 java.lang.NullPointerException: [E.IWP.WEBSERVICE.AXIS2.00006] 要素が null です。

SOAPボディ に *ユーザ情報* が存在しない、または ユーザ情報が格納されている要素名が規定の文字列（標準では `wsUserInfo`）ではない場合に発生します。

ユーザ情報 が格納されている要素名が `param0` 等の場合は、Webサービス として公開しているJavaクラスが `-g` オプションなしでコンパイルされている可能性があります。

公開するJavaクラスはデバッグ情報を生成するようにコンパイルしてください。これは SOAPボディ に含まれる ユーザ情報の名称を規定の文字列で取得するために必要な処理です。

> **コラム**
> デバッグ情報を生成するには `javac` コマンドのオプション `-g` を利用してコンパイルを行います。

> **コラム**
> *ユーザ情報* の要素名は *Webサービス・プロバイダ の認証・認可の設定* の *<wsUserInfoArgumentName>タグ* で設定します。

wsse:BadRequest

SOAP Fault Code	wsse:BadRequest
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00009] 指定された RequestSecurityToken を理解できません。
SOAP Fault Detail（発生時の例）	java.lang.IllegalStateException: [E.IWP.WEBSERVICE.AXIS2.00008] WSAuthModule が見つかりませんでした。 認証タイプ = PlainTextPassword

Webサービス・クライアント から送信された *ユーザ情報* の認証タイプに対応する認証モジュールが存在しない場合に発生します。

その他に、平文パスワード用認証モジュール を利用しない設定が行われている状態で認証タイプを未指定で Webサービス を実行した場合もこのエラーが発生します。

> **コラム**
>
> 平文パスワード用認証モジュール の利用可否は
> *Webサービス・プロバイダ の認証・認可の設定* の *<enablePlainTextPassword>タグ* で設定します。

> **コラム**
>
> 平文パスワード用認証モジュール 以外の対応する認証モジュールは
> *Webサービス・プロバイダ の認証・認可の設定* の *<authModule>タグ* で設定します。

wsse:AuthenticationBadElements

SOAP Fault Code	wsse:AuthenticationBadElements
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00010] ダイジェスト要素が不足しています。
SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.AuthenticationBadElementsException: [E.IWP.WEBSERVICE.AUTH.00016] WSSE認証文字列が不正です。 java.lang.IllegalStateException: No match found

Webサービス・クライアント にて指定された認証タイプにおける WSSE認証文字列 が間違っている場合に発生します。
例えば、認証タイプが `WSSE` の場合、送信された パスワード・ダイジェスト が WSSE認証文字列 として正しくない場合にこのエラーが発生します。
例） WSSE認証文字列 内にNonce項目が存在しない、等

> **コラム**
>
> WSSE認証文字列 の詳細については「*WSSE認証モジュール*」を参照してください。

wsse:ExpiredData

SOAP Fault Code	wsse:ExpiredData
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00003] 要求データが最新ではありません。
SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.ExpiredDataException: [E.IWP.WEBSERVICE.AUTH.00018] 有効期限が過ぎています。

Webサービス・クライアント から送信されたデータの期限を過ぎている場合に発生します。
例えば、認証タイプが `WSSE` の場合、システム日時と WSSE認証文字列 のCreatedを比較し、設定された期限が過ぎている場合にこのエラーが発生します。

> **コラム**
>
> WSSE認証文字列 の有効期限は
> *Webサービス・プロバイダ の認証・認可の設定* の *WSSE認証モジュール <expire>タグ* で設定します。

wsse:InvalidSecurityToken

SOAP Fault Code	wsse:InvalidSecurityToken
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00004] セキュリティ トークンが拒否されました。

SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.InvalidSecurityTokenException: [E.IWP.WEBSERVICE.AUTH.00018] 有効期限内で2回以上受信されました。 Created = 1966-12-01T12:34:56Z

Webサービス・クライアント から送信されたデータが既に受信済みの場合に発生します。
例えば、認証タイプが WSSE の場合、WSSE認証文字列 のCreatedとNonceを有効期限で設定された期間だけサーバ側で保持しています。既に保持されているCreatedとNonceで認証しようとした場合、このエラーが発生します。

> **コラム**
>
> WSSE認証文字列 の有効期限は
> *Webサービス・プロバイダ の認証・認可の設定* の *WSSE認証モジュール <expire>タグ* で設定します。

wsse:FailedAuthentication

SOAP Fault Code	wsse:FailedAuthentication
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00001] 認証に失敗しました。
SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.AuthenticationException: [E.IWP.WEBSERVICE.AUTH.00014] 認証に失敗しました。 ユーザCD = aoyagi ...

このエラーはいくつかの発生原因が存在します。以下にそれらの原因について記述します。

アカウントが登録されていません。

SOAP Fault Detail （発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.AuthenticationException: [E.IWP.WEBSERVICE.AUTH.00014] 認証に失敗しました。 ユーザCD = aoyagi jp.co.intra_mart.foundation.security.exception.AccessSecurityException: [E.IWP.WEBSERVICE.AUTH.00004] アカウントが登録されていません。 ユーザCD = aoyagi

ユーザ情報で指定されたユーザIDを持つアカウントが存在しない場合に発生します。ユーザIDが正しいことを確認してください。

アカウントライセンスが登録されていません。

SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.AuthenticationException: [E.IWP.WEBSERVICE.AUTH.00014] 認証に失敗しました。 ユーザCD = aoyagi jp.co.intra_mart.foundation.security.exception.AccessSecurityException: [E.IWP.WEBSERVICE.AUTH.00005] アカウントライセンスが登録されていません。 ユーザCD = aoyagi

ユーザ情報で指定されたユーザIDを持つアカウントのアカウントライセンスが無い場合に発生します。

アカウントの有効期限が切れています。

SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.AuthenticationException: [E.IWP.WEBSERVICE.AUTH.00014] 認証に失敗しました。 ユーザCD = aoyagi jp.co.intra_mart.foundation.security.exception.AccessSecurityException: [E.IWP.WEBSERVICE.AUTH.00006] アカウントの有効期限が切れています。 ユーザCD = aoyagi

ユーザ情報で指定されたユーザIDを持つアカウント有効期限が切れている場合に発生します。

アカウントがロックされています。

SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.AuthenticationException: [E.IWP.WEBSERVICE.AUTH.00014] 認証に失敗しました。 ユーザCD = aoyagi jp.co.intra_mart.foundation.security.exception.AccessSecurityException: [E.IWP.WEBSERVICE.AUTH.00007] アカウントがロックされています。 ユーザCD = aoyagi

ユーザ情報で指定されたユーザIDを持つアカウントがロックされている場合に発生します。

パスワードが間違っています。

SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.AuthenticationException: [E.IWP.WEBSERVICE.AUTH.00014] 認証に失敗しました。 ユーザCD = aoyagi jp.co.intra_mart.foundation.security.exception.AccessSecurityException: [E.IWP.WEBSERVICE.AUTH.00015] パスワードが間違っています。 ユーザCD = aoyagi

ユーザ情報で指定されたユーザIDを持つパスワードと指定されたパスワードが間違っている場合に発生します。

wsse:RequestFailed

SOAP Fault Code	wsse:RequestFailed
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00002] 指定した要求に失敗しました。
SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.web_service.auth.exception.AuthorizationException: [E.IWP.WEBSERVICE.AUTH.00001] アクセス権限がありません。 Webサービス名 = MenuService, オペレーション名 = getAvailbleMenuTree

指定された Webサービス・オペレーション を実行する権限がない場合に発生します。

> **コラム**
>
> 管理者による認可設定方法については、「*Webサービス・オペレーション への認可設定*」を参照してください。

wsse:TenantNotResolved

SOAP Fault Code	wsse:TenantNotResolved
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00018] 指定した要求に失敗しました。
SOAP Fault Detail（発生時の例）	java.lang.IllegalStateException: [E.IWP.WEBSERVICE.AUTH.00017] テナントIDを解決できません。

指定された Webサービス・オペレーション が呼び出された環境で、テナントIDが解決できない場合に発生します。

wsse:InvalidLoginGroupID

SOAP Fault Code	wsse:InvalidLoginGroupID
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00020] 要求が無効か、形式が間違っています。
SOAP Fault Detail（発生時の例）	java.lang.IllegalStateException: [E.IWP.WEBSERVICE.AUTH.00019] loginGroupID パラメータが null です。

ユーザ情報でログイングループIDが指定されていない場合に発生します。

wsse:TenantIdNotMatch

SOAP Fault Code	wsse:TenantIdNotMatch
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00022] 要求が無効か、形式が間違っています。
SOAP Fault Detail（発生時の例）	java.lang.IllegalStateException: [E.IWP.WEBSERVICE.AUTH.00021] 解決されたテナントID(secondary)と、ユーザ情報に指定されたテナントID(default)が一致しません。

指定された Webサービス・オペレーション が呼び出された環境で解決されたテナントIDと、ユーザ情報で指定されたログイングループIDが一致しない場合に発生します。

wsse:FailedTenantSwitch

SOAP Fault Code	wsse:FailedTenantSwitch
SOAP Fault Reason	[E.IWP.WEBSERVICE.AXIS2.00023] テナントID(secondary)の切り替えに失敗しました。
SOAP Fault Detail（発生時の例）	jp.co.intra_mart.foundation.admin.exception.AdminException: [E.IWP.ADMIN.TENANT.10057] ログインユーザはテナントの切り替えができません。

Webサービスを実行するための一時的なテナントの切り替えに失敗した場合に発生します。

項目

aarファイルを作成する

aarファイル（Axis2 Archiveファイル）を作成するツールについて紹介します。
ここでは、Antタスク「aarGenerate」を利用します。このAntタスクは、services.xml を作成し、services.xmlを含んだaarファイルを作成します。
aarファイルの構成は、Axis2 プロジェクトの「Apache Axis2 Web Administrator's Guide Step 3: Create Archive File」を参照してください。

この手順は以下の条件を満たしている環境で行うことを前提とします。

- Apache Ant がインストールされていること
- 「Webサービス用ツール」が存在すること

  「Webサービス用ツール」は プロダクトファイルダウンロード より <WebService_Tools.zip> をダウンロードしてください。
  ※ ダウンロードには製品のライセンスキーが必要です。

Webサービス用ツールを展開したディレクトリを %WEBSERVICE_TOOL_HOME% として以降記述します。

> **コラム**
>
> ビルドツール「Ant」がインストールされていない場合は以下のサイトを参考にしてインストールを行ってください。
>
> Apache Ant Manual - Installing Apache Ant

> **注意**
>
> このツールでは、jarやクラスをaarファイルに含めることはできません。aarファイルの Webサービス を実行するには、アプリケーションサーバのクラスパス上に Webサービス の実装クラスを配置する必要があります。
> 具体的には、jarファイルを %CONTEXT_PATH%/WEB-INF/lib 配下に、クラスを %CONTEXT_PATH%/WEB-INF/classes 配下に配置してください。

1. 公開する Webサービス 用の各種設定を行います。

   1. AarGen.propertiesの編集を行います。
      %WEBSERVICE_TOOL_HOME%/AarGen.propertiesの編集を行います。
      各プロパティの説明は以下の通りです。

プロパティ名	必須	説明
serviceName	○	Webサービス として公開する名称を指定します。この値はaarファイルの名称です。
className	○	Webサービス の実装クラスを指定します。クラスは完全修飾名で指定します。
destDir	○	aarファイルを出力するディレクトリを指定します。
moduleRef	×	Webサービス に適用する Axis2 モジュール 名を指定します。 ,（カンマ）を使用することで複数指定が可能です。
wsdlDir	×	aarファイルに含める WSDL やXSDが配置されているディレクトリ。 未指定の場合、services.xml以外のファイルをaarファイルに含めません。
serviceAuthzUri	×	Webサービス が提供するサービスに対して認可リソースをURIで指定します。 未指定の場合、サービスに対しての認可リソース設定を行いません。
opeAuthzUriFile	×	Webサービス・オペレーション への認可リソース設定が記述されたCSVファイルを指定します。 未指定の場合、Webサービス・オペレーション に対しての認可リソース設定を行いません。
verbose	×	Ant タスク実行時に詳細情報を出力するかを指定します。

注意

Windows環境でパスを指定する場合は区切り文字を / または \ としてください。

2. opeAuthzUriFile.csvの編集を行います。
   Webサービス が提供する Webサービス・オペレーション に対して認可リソースをURIで指定します。
   %WEBSERVICE_TOOL_HOME%/opeAuthzUri.csvを編集します。（AarGen.propertiesの `opeAuthzUriFile` で指定した値のファイルを編集します。）
   Webサービス・オペレーション に対して認可リソースを指定しない場合は、このファイルを編集する必要はありません。
   このファイルはCSV形式で記述します。フィールド区切り文字は ,（カンマ）、レコード区切り文字は改行です。
   以下の例の場合、
   - Webサービス・オペレーション `add` に対して URI `service://sample/web_service/member_info/add`
   - Webサービス・オペレーション `find` に対して URI `service://sample/web_service/member_info/find`
   - Webサービス・オペレーション `findAll` に対して URI `service://sample/web_service/member_info/findAll`

   です。

```
add,service://sample/web_service/member_info/add
find,service://sample/web_service/member_info/find
findAll,service://sample/web_service/member_info/findAll
```

コラム

サービス と Webサービス・オペレーション の両方に認可リソースが指定されている場合は、Webサービス・オペレーション に指定されている認可リソースが優先されます。
詳細は「*services.xmlに認可リソースを指定する*」を参照してください。

> **注意**
>
> 認可リソースを指定した場合、指定したURIに該当する認可リソースが存在しないと Webサービス・オペレーション 実行時に以下のようなエラーが発生します。
>
> ```
> [E.IWP.AUTHZ.DECISION.10007] リソースグループが登録されていません。
> ```
>
> あらかじめ認可リソースを作成の上、Webサービス・オペレーション を実行してください。
>
> 認可リソースの追加方法については「テナント管理者操作ガイド」-「認可を設定する」の「リソースを追加する」を参照してください。

3. AarGenを実行するための環境情報を指定します。
   Apache Antがインストールされているディレクトリを指定します。
   %WEBSERVICE_TOOL_HOME%/AarGen.bat（Unix系OSの場合はAarGeb.sh）を編集します。
   環境変数「ANT_HOME」に対してAntがインストールされているディレクトリを指定します。
   （Windows系OSの場合）

   ```
   REM AarGen.bat
   set ANT_HOME=C:/apache-ant
   ```

   （Unix系OSの場合）

   ```
   # AarGen.sh
   export ANT_HOME=/apache-ant
   ```

2. AarGenを実行します。
   同梱されているバッチファイルを実行します。

   （Windows系OSの場合）

   ```
   > %WEBSERVICE_TOOL_HOME%\AarGen.bat
   ```

   （Unix系OSの場合）

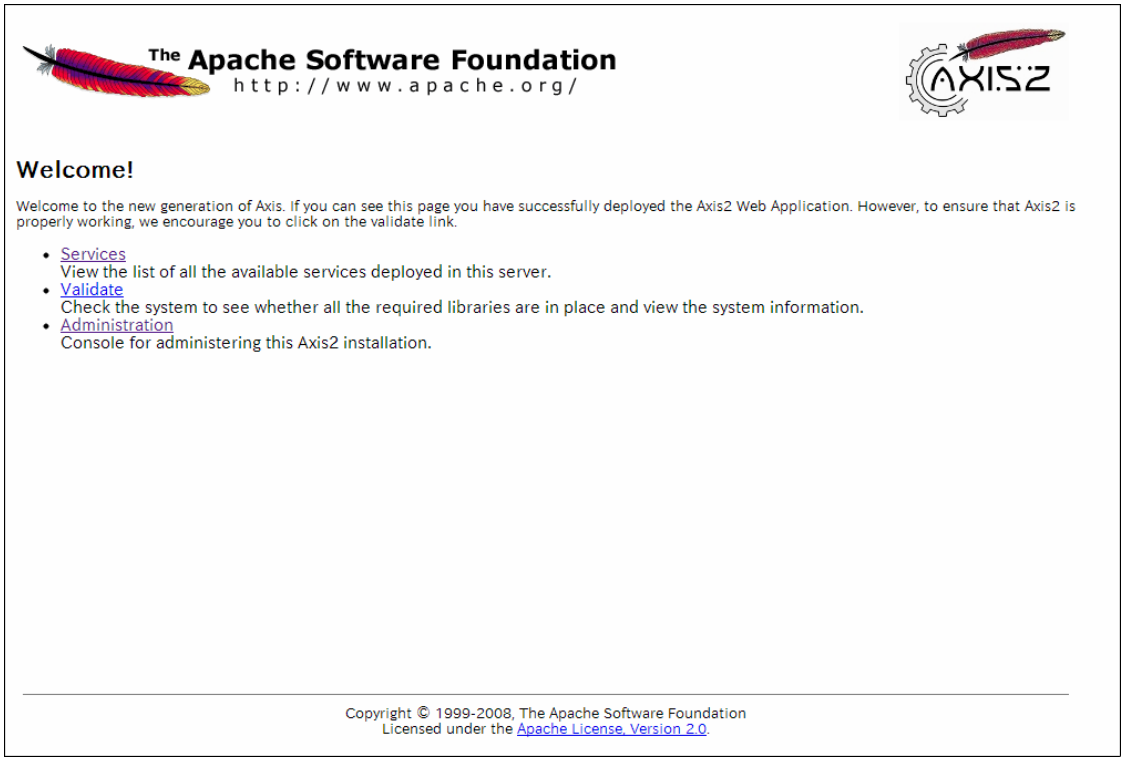
   ```
   $ sh %WEBSERVICE_TOOL_HOME%/AarGen.sh
   ```

   AarGen.propertiesの `destDir` で指定されているディレクトリの配下にaarファイルが出力されます。

Apache Axis2 管理コンソールについて

> **注意**
>
> Apache Axis2 管理コンソールは、 intra-mart Accel Platform 2020 Winter(Azalea) 以降の環境で使用できません。

Apache Axis2 から提供されている管理用コンソールです。
http://<HOST>:<PORT>/<CONTEXT_PATH>/axis2-web/index.jsp でアクセスできます。
この管理コンソールを利用することで、ブラウザ経由でデプロイされている Webサービス の情報の閲覧や、Webサービス のデプロイが可能です。



Apache Axis2 管理コンソールの詳細については「Apache Axis2 Web Administrator's Guide」を参照してください。

> **コラム**
> 標準のユーザ名は admin、 パスワードは axis2 です。この値は %CONTEXT_PATH%/conf/axis2.xml で変更可能です。

services.xmlについて

Apache Axis2 には、Webサービス名や Webサービス として公開するJavaクラスなどのサービス情報を設定するための「services.xml」があります。

このファイルを規定のフォルダに配置したり、aarファイルに配置したりすることで Webサービス のデプロイが可能です。
Webサービス のデプロイについては、「*Webサービス をデプロイする*」を参照してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- サービスの設定を行います。Webサービス名がSampleServiceであることを指定しています。 -->
<service name="SampleService">
   <!-- Webサービスに対しての説明です。 -->
   <description>The Web service on the intra-mart.</description>

   <!-- このサービスのServiceClassがsample.web_service.provider.SampleServiceであることを指定しています。 -->
   <parameter name="ServiceClass">sample.web_service.provider.SampleService</parameter>

   <!-- このサービスのWebサービス・オペレーション実行時に認証・認可を行うことを指定しています。 -->
   <module ref="im_ws_auth"/>

   <!-- このサービスに対してのメッセージレシーバを指定しています。 -->
   <messageReceivers>
      <!-- 返却値が無いメソッドに対してのレシーバを指定しています。 -->
      <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
      <!-- 何らかの返却を行うメソッドに対してのレシーバを指定しています。 -->
      <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
   </messageReceivers>

</service>
```

<parameter> タグの name 属性の属性値を ServiceClass と指定し、タグ内部に Webサービス の実装クラスのJava完全修飾名を指定します。これにより、指定したクラスのpublicメソッドが Webサービス・オペレーション として公開されます。

各種メッセージレシーバは、XMLの Webサービス メッセージとJavaのオブジェクトのマッピングを行うためのものです。
上記以外にも、「RowXMLxxxxMessageReceiver」というXMLを直接扱うためのメッセージレシーバも提供されています。

services.xmlの詳細は、Axis2プロジェクトの「Service Configuration」を参照してください。
メッセージレシーバの詳細は、Axis2のAPIドキュメントを参照してください。

services.xmlに認可リソースを指定する

Webサービス・オペレーション に対して認可リソースを指定します。認可リソースはURI形式で指定します。

認可リソースのURIについては「認可仕様書 リソース管理」を参照してください。

認可リソースは、サービスまたは、オペレーションに対して設定することが可能です。
認可リソースはservices.xmlの `<parameter>` タグを利用して指定します。`<parameter name="auth-uri">` の内部に認可リソースのURIを指定します。

```
<?xml version="1.0" encoding="UTF-8"?>
<service name="SampleService">
   <description>The Web service on the intra-mart.</description>

   <parameter name="ServiceClass">sample.web_service.provider.SampleService</parameter>

   <module ref="im_ws_auth"/>

   <messageReceivers>
      <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
      <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
   </messageReceivers>

   <!-- サービスSampleServiceに対して認可リソースを指定します。 -->
   <parameter name="authz-uri">service://sample/web_service/sample</parameter>
   <operation name="foo">
      <!-- サービスSampleServiceのオペレーションfooに対しての認可リソースを指定します。 -->
      <parameter name="authz-uri">service://sample/web_service/sample/foo</parameter>
   </operation>
   <operation name="bar">
      <!-- サービスSampleServiceのオペレーションbarに対しての認可リソースを指定します。 -->
      <parameter name="authz-uri">service://sample/web_service/sample/bar</parameter>
   </operation>
   <!--
   <operation name="baz">
      <parameter name="authz-uri">service://sample/web_service/sample/baz</parameter>
   </operation>
   -->

</service>
```

サービスとオペレーションの両方に認可リソースが指定された場合は、オペレーションに指定された認可リソースが有効です。上記の例の場合、オペレーション `bar` に対しての認可リソースURIは `service://sample/web_service/sample/bar` （オペレーションの認可リソースを利用）、オペレーション `baz` に対しての認可リソースURIは `service://sample/web_service/sample` （サービスの認可リソースを利用）です。

> **注意**
>
> サービスとオペレーションの両方とも認可リソースが未指定の場合、そのオペレーションの認可は常に「許可」です。

> **注意**
>
> 認可は Axis2 モジュール 「im_ws_auth」が適用されている Webサービス に対してのみ有効です。
> Axis2 モジュール 「im_ws_auth」の適用方法については「*Axis2 モジュール 「im_ws_auth」の適用方法について*」を参照してください。

注意

認可は Webサービス・プロバイダ の認可の設定が有効であるときのみ動作します。
詳細は、「*Webサービス・プロバイダ の認証・認可の設定*」の「*<enableAuthorization>タグ*」を参照してください。

注意

認可リソースを指定した場合、指定したURIに該当する認可リソースが存在しないと Webサービス・オペレーション 実行時に以下のようなエラーが発生します。

```
[E.IWP.AUTHZ.DECISION.10007] リソースグループが登録されていません。
```

あらかじめ認可リソースを作成の上、Webサービス・オペレーション を実行してください。

認可リソースの追加方法については「テナント管理者操作ガイド」-「認可を設定する」の「リソースを追加する」を参照してください。

コラム

URIについては以下の形式を推奨しています。

- サービスに対して設定する場合

  `service://%アプリケーションID%/web_service/%サービス名%`

- Webサービス・オペレーション に対して設定する場合

  `service://%アプリケーションID%/web_service/%サービス名%/%オペレーション名%`

Webサービス をデプロイする

Webサービス のデプロイ方法について紹介します。

ディレクトリ形式のデプロイ

%CONTEXT_PATH%/WEB-INF/services ディレクトリ配下に Webサービス の資材が格納された任意の名称のディレクトリを配置することで Webサービス をデプロイすることが可能です。
このディレクトリ配下に、Webサービス に関連するライブラリ、クラス、プロパティ・ファイル、WSDL ファイル、XSDファイルおよび、services.xmlを格納することで Webサービス がデプロイされます。

「services.xml」を含めたパスは以下の通りです。

%CONTEXT_PATH%/WEB-INF/services/%任意の名称%/META-INF/services.xml

aarファイル形式のデプロイ

aarファイル（Asix2 Archiveファイル）を、%CONTEXT_PATH%/WEB-INF/services ディレクトリに配置することで、Webサービスをデプロイすることが可能です。

aarファイルとは、Webサービス に関連するライブラリ、クラス、プロパティ・ファイル、WSDL ファイル、XSDファイルおよび、services.xmlを格納したファイルです。
aarファイルの作成については「*aarファイルを作成する*」を参照してください。
aarファイルの構成は、Axis2 プロジェクトの「Apache Axis2 Web Administrator's Guide Step 3: Create Archive File」を参照してください。

Axis2 モジュール 「im_ws_auth」の適用方法について

Webサービス を認証・認可の対象とする方法を紹介します。

Webサービス に対して認証・認可の対象とするには Axis2 モジュール 「im_ws_auth」を適用する必要があります。
Axis2 モジュール 「im_ws_auth」を適用する方法としては以下が存在します。

- *services.xml で Axis2 モジュール 「im_ws_auth」を指定する*
- *axis2.xml で Axis2 モジュール 「im_ws_auth」を指定する*
- *Apache Axis2 管理コンソールで Axis2 モジュール 「im_ws_auth」を指定する*

services.xmlで Axis2 モジュール 「im_ws_auth」を指定する

<service> タグ内部に <module ref="im_ws_auth"/> を記述することで Axis2 モジュール 「im_ws_auth」を指定することが可能です。
<operation> タグ内部に記述することで Webサービス・オペレーション 単位で指定することも可能です。

詳細は「*services.xmlについて*」を参照してください。

axis2.xmlで Axis2 モジュール 「im_ws_auth」を指定する

%CONTEXT_PATH%/WEB-INF/conf/axis2.xml を編集します。
以下の <service> タグのコメントアウトを除去します。
<service> タグに Axis2 モジュール 「im_ws_auth」を適用したい Webサービス 名を指定します。
指定したいサービスが複数ある場合は <service> タグを複数記述します。

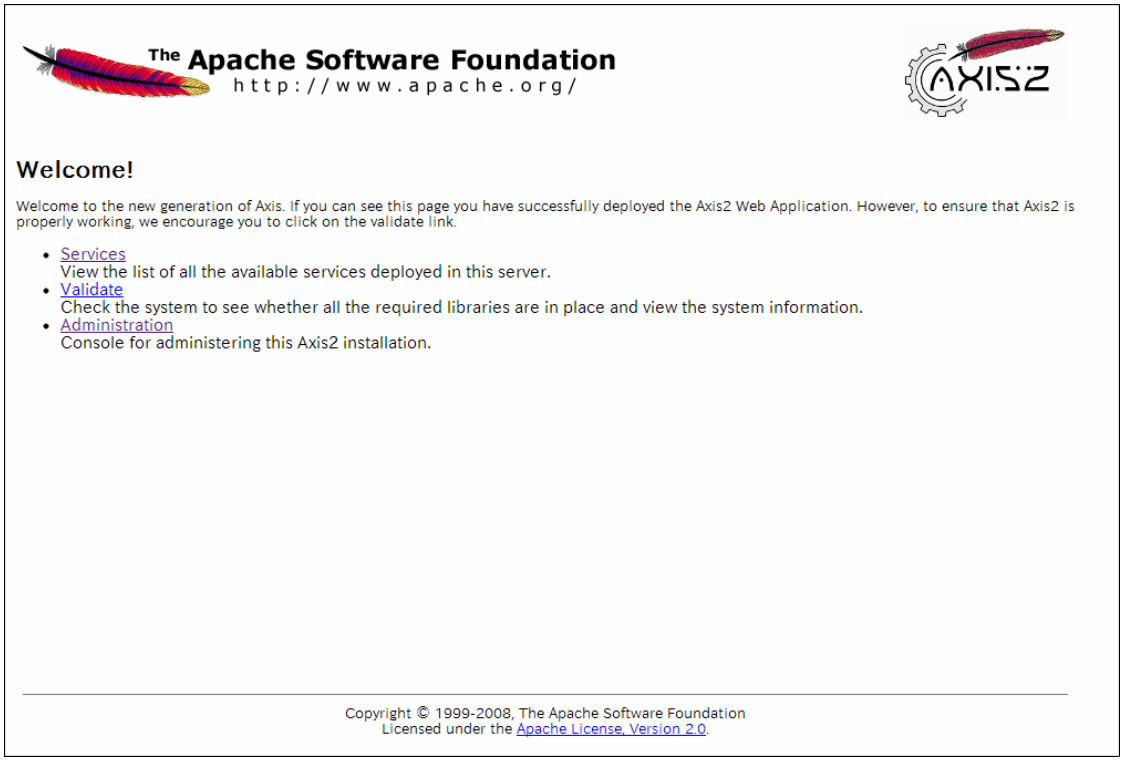
```
<listener class="jp.co.intra_mart.foundation.web_service.axis2.observers.EngageModuleAxisObserver">
  <parameter name="engageModule">
    <module ref="im_ws_auth">
      <!-- <service>ExampleWebServiceName</service> -->
    </module>
  </parameter>
</listener>
```

Apache Axis2 管理コンソールで Axis2 モジュール 「im_ws_auth」を指定する

> **注意**
>
> Apache Axis2 管理コンソールは、 intra-mart Accel Platform 2020 Winter(Azalea) 以降の環境で使用できません。

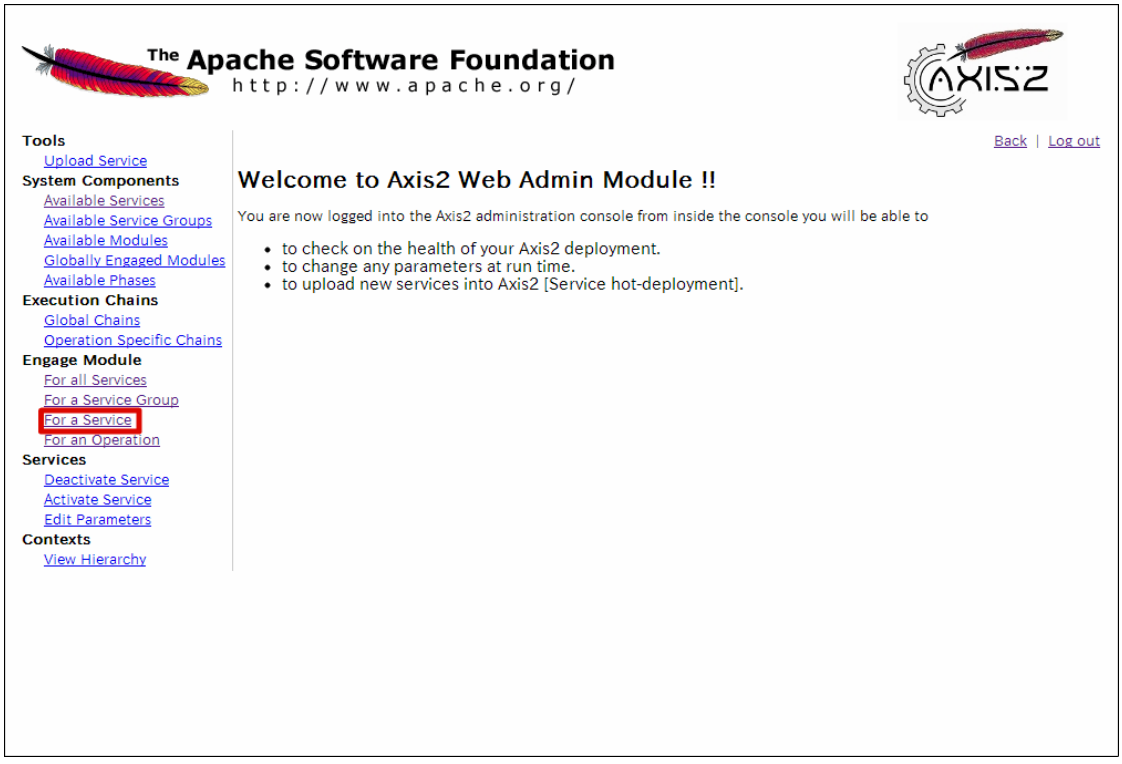
Apache Axis2 の管理コンソール機能を利用して Axis2 モジュール 「im_ws_auth」を適用することも可能です。
管理コンソールについては「*Apache Axis2 管理コンソールについて*」を参照してください。

1. 稼働中のサーバに対して以下にアクセスします。
   http://<HOST>:<PORT>/<CONTEXT_PATH>/axis2-web/index.jsp

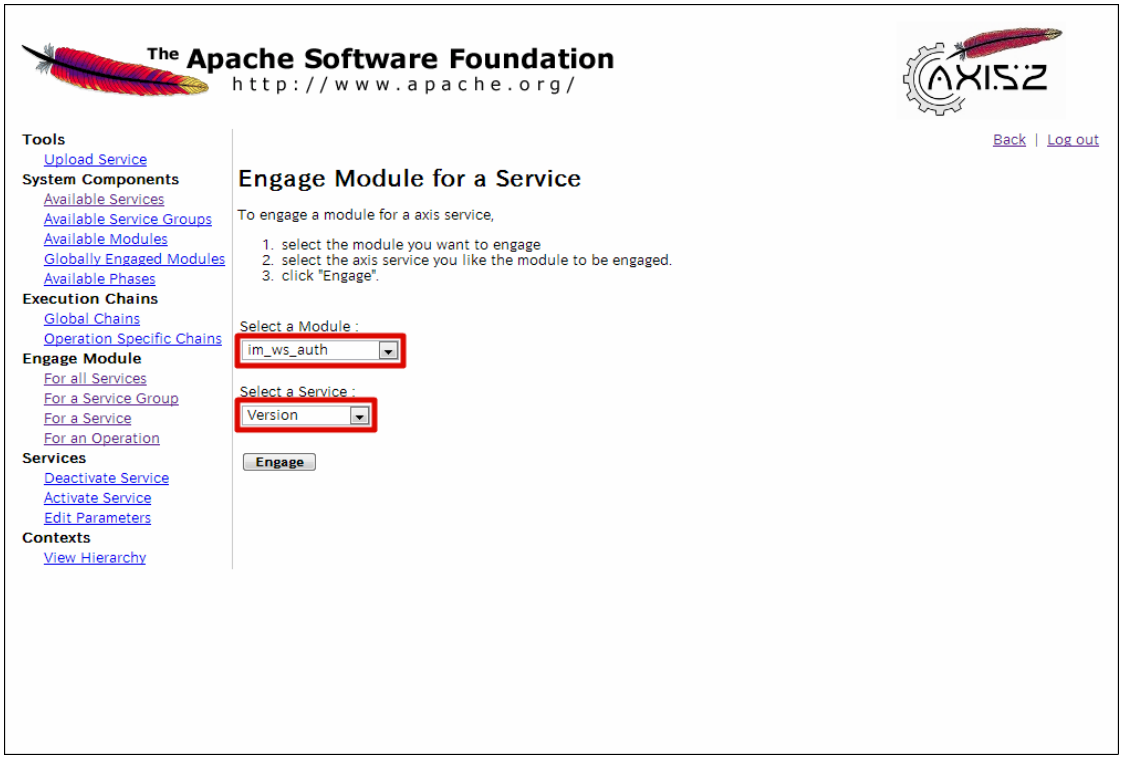


2. 「Administration」をクリックします。

3. Engage moduleの「For a service」をクリックします。



4. Select a Module: で「im_ws_auth」を、Select a Service: で任意のサービスを選択します。



5. 「Engage」をクリックします。
6. 上記で指定したサービスに対して Axis2 モジュール 「im_ws_auth」が適用されます。

> **注意**
> 管理コンソールを利用した Axis2 モジュール の設定はアプリケーションサーバの再起動を行うと初期化されます。

分散環境での WSDL について

Apache Axis2 が自動生成する WSDL に記述されているエンドポイントは、クラスタが稼働しているホスト名に依存します。
このため、分散環境時に WSDL の問い合わせを行うとWebサーバによってディスパッチされたアプリケーションサーバのホストがエンドポイントとして取得されてしまいます。
これを回避するためには、以下のいずれかの手段を取ります。

- WSDL の自動生成が行われないようにする

  ディレクトリ形式のデプロイ を行った場合、あらかじめWebサーバ経由のエンドポイントが記述された WSDL を作成し、`%CONTEXT_PATH%/WEB-INF/services/%任意の名称%/META-INF/%Webサービス名%.wsdl` に配置します。

- Webサービス・クライアント が指定するエンドポイントをWebサーバ経由のものに変更する

  > **コラム**
  > スクリプト開発モデル API SOAPClient を利用している場合は、SOAPClientの第４引数のエンドポイントに指定しているURLをWebサーバ経由のものに変更します。

  > **コラム**
  > WSDL からスタブを生成している場合は、スタブのコンストラクタで指定するエンドポイントに指定しているURLをWebサーバ経由のものに変更します。

認証モジュールを追加する

認証モジュールを追加するには、axis2.xmlに *<authModule>タグ* を追加します。
axis2.xml については「*Webサービス・プロバイダ の認証・認可の設定*」を参照してください。
以下は標準では設定されていない *未認証ユーザ用認証モジュール*
（`jp.co.intra_mart.foundation.web_service.auth.impl.WSAuthModule4Anonymous`） を認証モジュールとして追加する場合の設

定です。

```
<axisconfig name="AxisJava2.0">

    <!-- ================================================= -->
    <!-- Parameters for intra-mart -->
    <!-- ================================================= -->
    <parameter name="jp.co.intra_mart.foundation.web_service">

        <enablePlainTextPassword>false</enablePlainTextPassword>

        <!-- authModuleタグ追加 -->
        <authModule class="jp.co.intra_mart.foundation.web_service.auth.impl.WSAuthModule4Anonymous" />

        <authModule class="jp.co.intra_mart.foundation.web_service.auth.impl.WSAuthModule4WSSE">
            <expire>300</expire>
        </authModule>

        <enableAuthentication>true</enableAuthentication>
        <enableAuthorization>true</enableAuthorization>

        <showSoapFaultDetail>true</showSoapFaultDetail>
        <wsUserInfoArgumentName>wsUserInfo</wsUserInfoArgumentName>

    </parameter>

    <!-- 省略 -->

</axisconfig>
```

SOAP メッセージのモニタリング

Webサービス 実行時にやり取りされる Webサービス メッセージをモニタリングする方法を紹介します。

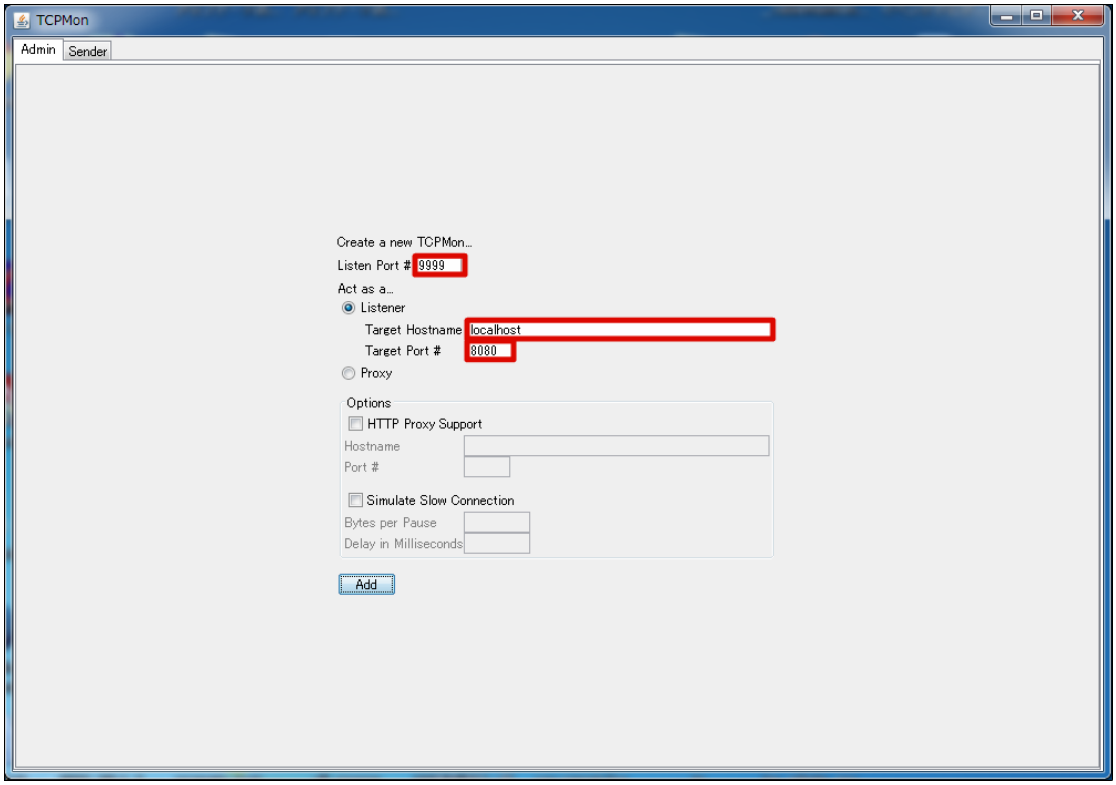
TCPMonによるモニタリング

Apache Web Services Projectで提供されているメッセージモニタリングツール「TCPMon」の利用方法を紹介します。

TCPMonの詳細は、「TCPMon」を参照してください。

1. TCPMonのダウンロードと解凍
   「TCPMonのダウンロードページ」からTCPMonをダウンロードし、アーカイブを解凍します。このディレクトリを%TCPMON_HOME% とします。

2. TCPMonの起動と設定
   %TCPMON_HOME%/build/tcpmon.bat（またはtcpmon.sh）を実行し、TCPMonを起動します。
   「Admin」タブを表示し、Listen Port に適当なポート番号を設定します。ポート番号は他のサービスと重複しないポート番号を指定してください。
   本書では例として「9999」とします。
   Target Hostname と Target Port にWebサービスがデプロイされているホストの情報を設定します。
   本書では例としてTarget Hostnameを「localhost」、Target Portを「8080」とします。
   その後、「Add」をクリックします。

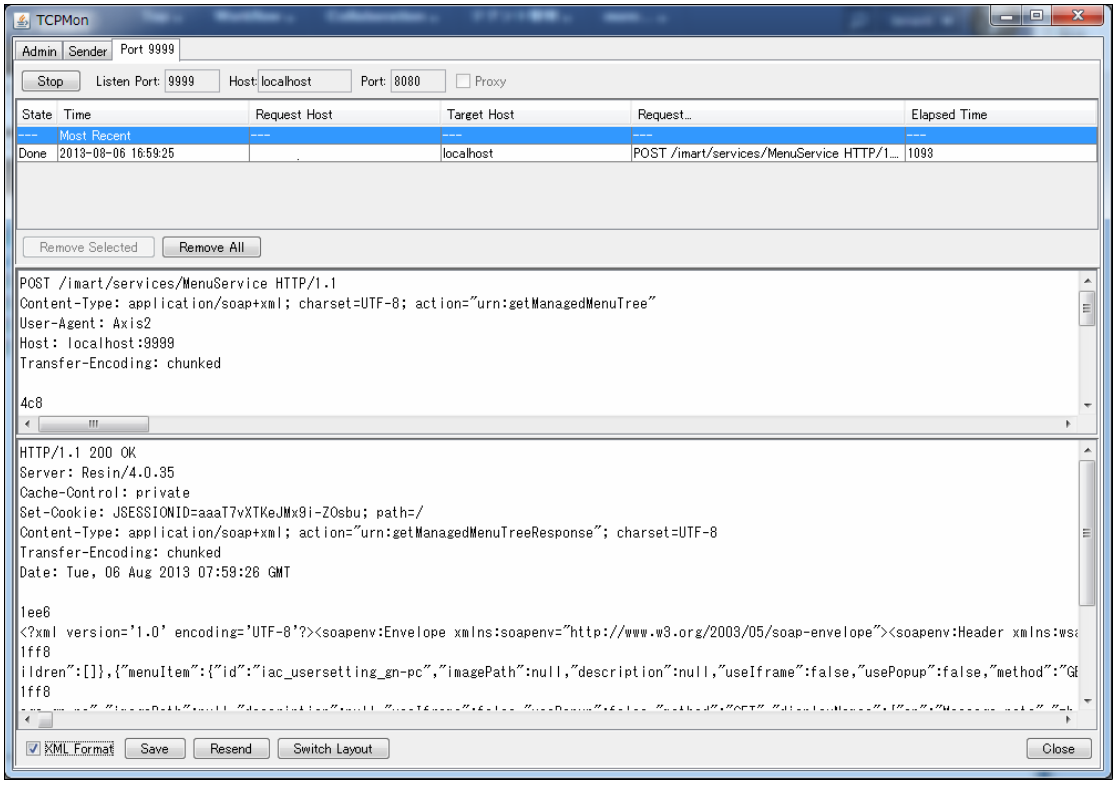


3. Webサービス・クライアント でエンドポイントを変更する
   Webサービス・クライアント が実行時に指定するエンドポイントのポート番号を先ほど指定した Listen Port に変更します。

> **コラム**
> スクリプト開発モデル API SOAPClient を利用している場合は、SOAPClientの第４引数のエンドポイントに指定しているポート番号を Listen Port のポート番号に変更します。

> **コラム**
> WSDL からスタブを生成している場合は、スタブのコンストラクタで指定するエンドポイントに指定しているポート番号を Listen Port のポート番号に変更します。

4. 「Port <Listen Port のポート番号>」タブから SOAP メッセージを閲覧する



認証・認可を必要としない Webサービス について

認証、認可およびログインを必要としない Webサービス を作成する場合、Axis2 モジュール 「im_ws_auth」を適用する必要がありません。
具体的には、services.xmlの<module ref="im_ws_auth"/>を設定しないことで、該当モジュールが適用されなくなります。

Axis2 モジュール 「im_ws_auth」を適用しないこと以外は、本書で説明している Webサービス と同じ方法で作成することが可能です。
Axis2 モジュール 「im_ws_auth」の適用については「*Axis2 モジュール 「im_ws_auth」の適用方法について*」を参照してください。

コラム
各 Webサービス・オペレーション の引数に *ユーザ情報* を受け取る必要もありません。