Table of Contents

Revision Information

Revision Date	Updated Contents
2013-10-01	Initial Version
2014-04-01	Version 2 Following additions and changes have been made. • Added a column to [***Implementing Web Service Client***].

Introduction

Document Purpose

This document describes how to create and provide Web Service on intra-mart Accel Platform in Java language.

Web Service Java Development Programming Guide
VERSION 2 2014-04-01

Scope of descriptions are as follows :

- Method of creating Web Service Provider and sample descriptions
- Method of creating Web Service Client and sample descriptions

Target Readers

Following are the target readers of this document :

- Those who understand Web Service on intra-mart Accel Platform
- Developers of application program which provides Web Service Provider
- Developers of application program which uses Web Service Client

Remarks

1. There are several limitations in using Web Service. For more information about these limitations, please refer to [**Release Note Limitations**].
2. Part of the material for the tutorial described in this document is also covered by [**Web Service Script Development Programming Guide**]. If the material created in this document and the material created in “Web Service Script Development Programming Guide” are deployed at the same time, sample program may not work.

Document Structure

- ***Overview***

  This section provides summary information of Web Service in Java.

- ***Creating Web Service Provider***

  This section describes how to deploy Web Service on intra-mart Accel Platform using Java.

- ***Creating Web Service Client***

  This section describes how to utilize Web Service which is deployed on intra-mart Accel Platform using Java.

- ***Appendix***

  This section describes problems and resolutions related to the development of Web Service, and also includes additional sets of sample code.

Overview

Topics

About Web Service

In this document [Web Service] means Web Service that utilizes SOAP and WSDL.
For details about Web Service, please refer to [**Web Service Authentication and Authorization Specifications**].

Remarks about the use of Tutorial in this Document

- e Builder is used in this tutorial.

  e Builder will be used to develop and deploy the materials created in this tutorial.
  Please make it available beforehand.

- General Java knowledge is required.

  General Java knowledge for the creation of Web Service Provider and Web Service Client is required.

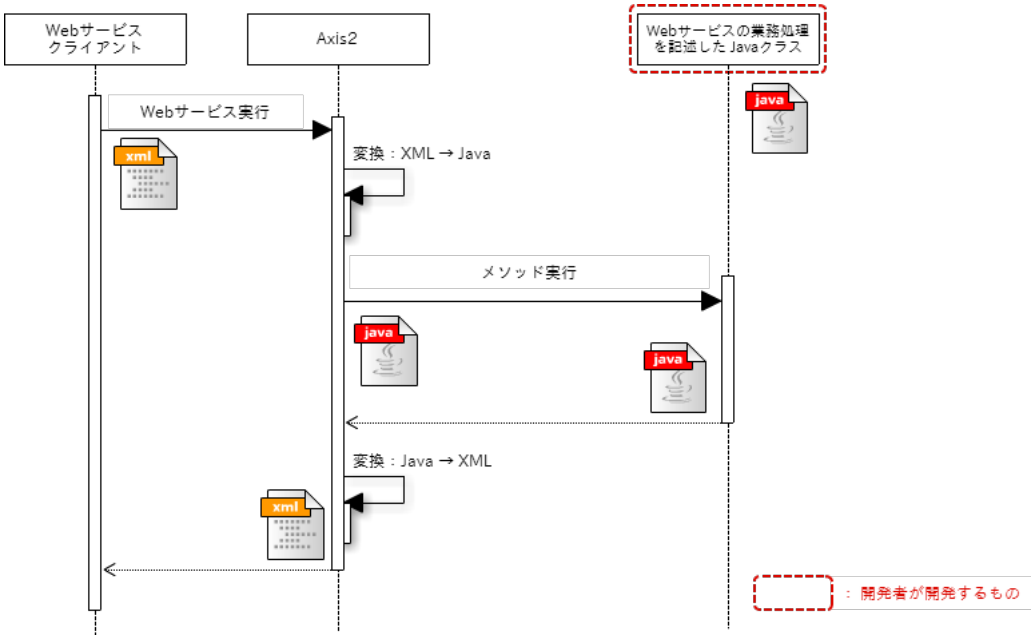
Creating Web Service Provider

Topics

Overview of Creation Steps

This chapter describes the procedures for publishing as Web Service.
Java class which is published as Web Service will be created.

Flow throughout the execution of Java class publsihed as Web Service is shown below :



1. Web Service Client requests the execution of Web Service.
2. Web service execution engine [Apache Axis2] calls the method of Java class which corresponds to the accepted request.
3. Execution result is returned to Web Service Client.

Preparing the Deployment of Web Service

Preparing the Development Environment

First of all, development environment for the creation of necessary materials to provide Web services should be prepared.
In this tutorial, e Builder is used to create the project below, and the procedures for development are described.

Group ID	mypackage (default setting is used)
Version	1.0.0 (default setting is used)
Project Name	sample_provider

Please start by installing e Builder, Resin, and intra-mart Accel Platform, and build the development environment.
For the installation procedures, please refer to [**e Builder Setup Guide**].

Warning

During the installation of intra-mart Accel Platform, please select [Web Service Authentication and Authorization] module and [Web Service Authentication and Authorization Client] module from the Base Module.
Unless these selections are made, compilation error would result for the Java code in the tutorial.

Upon the completion of e Builder installation, please create the project by [Module Project] following the steps below :
For more information, please refer to [**Module Project Creation in e Builder Users Operations Guide**].

1. Click [File] - [New] - [Project].
2. Select [e Builder] - [Module Project], and click [Next].
3. Enter [sample_provider] as a project name, and click [Finish].

Once the project creation is completed, project settings should be made to make APIs of intra-mart Accel Platform available.
For more information, please refer to [**Basic Functions of Module Development in e Builder Users Operations Guide**].

1. Right click on the project, and select [Property].
2. Select [e Builder]-[Module Assembly].
3. Select the folder with the same name as the context path that has been generated by deploying the war to the Web archive directory.
4. Turn off all the checkboxes in the list of automatic deployment destination for the resource change.
5. Click [OK].

Resolving Dependencies

Once the project settings are completed, please proceed to the modifications of dependencies.
You need to rely on the [Web Service Authentication and Authorization] module to create Web Service Provider.

Modify the project dependencies by following the steps below :

1. Double click on module.xml which is placed in the root directory of created project.
2. Open the [Dependency] tab, and click [Add].
3. Enter the contents below, and click [OK].

ID	jp.co.intra_mart.im_ws_auth
Version	8.0.2

Note

Please basically specify the version number which is supported.
If the APIs you want to use are included in the other version, please specify that version number.

4. After module.xml file is saved, please open [module.xml] tab, and remove the unnecessary tags (<tags>).
   Source code would finally end up with this :

```
<?xml version="1.0" encoding="UTF-8"?>
<module conf:schemaLocation="urn:intramart:jackling:toolkit:configurations configurations.xsd"
        xmlns="urn:intramart:jackling:module" xmlns:conf="urn:intramart:jackling:toolkit:configurations"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemeLocation="urn:intramart:jackling:module module.xsd">

    <id>mypackage.sample_provider</id>
    <version>1.0.0</version>
    <type>module</type>
    <name>${module.name}</name>
    <vendor>${module.vendor}</vendor>
    <description>${module.description}</description>

    <dependencies>
        <dependency>
            <module-id>jp.co.intra_mart.im_ws_auth</module-id>
            <verified-version min="8.0.2">8.0.2</verified-version>
        </dependency>
    </dependencies>
</module>
```

5. Open [Dependency Layer] tab. It is successful if dependencies are resolved and various modules are displayed.

Creating Business Processes in Java

After the completion of project preparation, please create the Java class to be published as Web Service and state the business processes.

In the sample web services created here, member information in the format below will be saved and searched.

Property	Description (Type)
id	Member ID (String)
name	Member Name (String)
age	Age (Integer)
married	true if married, false if not married (Boolean).
birthDate	Date of Birth (Date)
children	Children Information (Array in member information format)

Creating Type Information Class

Create the JavaBean to be handled by Web Services.

Type information class should be created by e Builder following the steps below :

1. Right click on the project, and select [New]-[Class].
2. Enter the contents below, and click [OK].

Package	sample.web_service.provider
Name	Member

3. Member.java is created under src/main/java in the project.
4. In order to retain each property of member information, the private variable and the access method are defined.

```java
package sample.web_service.provider;

import java.io.Serializable;
import java.util.Date;

public class Member implements Serializable {

    // 用途に応じて、変更する必要がある場合は変更してください。
    private static final long serialVersionUID = 1L;

    private String id;

    private String name;

    private Integer age;

    private Boolean married;

    private Date birthDate;

    private Member[] children;

    public Integer getAge() {
        return age;
    }

    public Date getBirthDate() {
        return birthDate;
    }

    public Member[] getChildren() {
        return children;
    }

    public String getId() {
        return id;
    }

    public Boolean getMarried() {
        return married;
    }

    public String getName() {
        return name;
    }

    public void setAge(final Integer age) {
        this.age = age;
    }

    public void setBirthDate(final Date birthDate) {
        this.birthDate = birthDate;
    }

    public void setChildren(final Member[] children) {
        this.children = children;
    }

    public void setId(final String id) {
        this.id = id;
    }

    public void setMarried(final Boolean married) {
        this.married = married;
    }

    public void setName(final String name) {
        this.name = name;
    }

}
```

Note

Since this sample handles member information by Permanent API, `Serializable` interface is implemented in `Member` class.

Warning

You should not use the classes with inheritence relationship for arguments or return values of methods published as Web Services.
If any class with inheritence relationship is used, error may occur on the client side of Web Services.
This limitation comes from the ADB(Axis Data Binding) specifications where XML name space is integrated by subclasses when Java objects are converted to XML.

For example, if the SubModel below is defined as a child class of the ParentModel, you are not allowed to use the SubModel as an argument or return value of the method that is published as a Web Service.

- sample.foo.ParentModel
- sample.bar.SubModel

Implementing the Web Service Processes

Please prepare `sample.web_service.provider.MemberInfoOperatorService.java`.
Publish the methods [add()], [find()], and [findAll()] defined in this class as Web Services.

Follow the steps below to create the Java class published as Web Service by e Builder.

1. Right click on the project, and select [New]-[Class].
2. Enter the contents below, and click [OK].

Package	sample.web_service.provider
Name	MemberInfoOperatorService

3. MemberInfoOperatorService.java is created under `src/main/java` in the project.
4. Implement MemberInfoOperatorService.java.

   Source code for MemberInfoOperatorService.java is as follows :

```
package sample.web_service.provider;

import java.io.IOException;
import java.util.List;

import jp.co.intra_mart.foundation.service.client.information.PermanentDirectory;
import jp.co.intra_mart.foundation.service.client.information.TreasureFile;
import jp.co.intra_mart.foundation.web_service.auth.WSUserInfo;

import org.apache.axis2.AxisFault;

public class MemberInfoOperatorService {

    private static final String DOMAIN = "sample_web_service";

    private static final String GROUP = "sample_member_info";

    public Boolean add(final WSUserInfo wsUserInfo, final Member member) throws AxisFault {
        final TreasureFile<Member> treasure = getPermanentFile();
        try {
            treasure.put(member.getId(), member);
            return Boolean.TRUE;
        } catch (final IOException e) {
            throw AxisFault.makeFault(e);
        } catch (final ClassNotFoundException e) {
            throw AxisFault.makeFault(e);
        }
    }

    public Member find(final WSUserInfo wsUserInfo, final String id) throws AxisFault {
        final TreasureFile<Member> treasure = getPermanentFile();
        try {
            final Member member = treasure.get(id);
            return member;
        } catch (final IOException e) {
            throw AxisFault.makeFault(e);
        } catch (final ClassNotFoundException e) {
            throw AxisFault.makeFault(e);
        }
    }

    public Member[] findAll(final WSUserInfo wsUserInfo) throws AxisFault {
        final TreasureFile<Member> treasure = getPermanentFile();
        try {
            final List<String> keyList = treasure.keyList();
            final int size = keyList.size();
            final Member[] members = new Member[size];
            for (int i = 0; i < size; i++) {
                members[i] = treasure.get(keyList.get(i));
            }
            return members;
        } catch (final IOException e) {
            throw AxisFault.makeFault(e);
        } catch (final ClassNotFoundException e) {
            throw AxisFault.makeFault(e);
        }
    }

    private TreasureFile<Member> getPermanentFile() {
        return PermanentDirectory.getInstance(DOMAIN).getFile(GROUP);
    }

}
```

Note

Permanent API should be used to save the member information.

Note

By throwing AxisFault, contents thrown can be sent to Web Service Client as a response.
For more information about AxisFault, please refer to [**Axis2 API Document**].

Creating services.xml

After the completion of Web Service implementation class, please make the setting file [services.xml] of Web Service available for use.

Create the services.xml file at the location in the project below :

- src/main/webapp/WEB-INF/services/sample_member_info/META-INF/services.xml

Source code of services.xml is as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceGroup>
  <service name="SampleMemberInfoOperatorService">
    <parameter name="ServiceClass">sample.web_service.provider.MemberInfoOperatorService</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>

    <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>

    <operation name="add">
      <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
    </operation>

    <operation name="find">
      <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
    </operation>

    <operation name="findAll">
      <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
    </operation>
  </service>
</serviceGroup>
```

Note

[Resource URI] for authorization should be set to set access authorities for each function.
Resource URI that is accessible by any user regardless of the authentication completion status should be initially allocated to check the sample behaviors.

- service://intra-mart.jp/public-resources/welcome-to-intramart

For the use of Web Service for the actual business purpose, please prepare the individual [Resource URI].
For details about authority settings, please refer to [***Setting Access Authority by using Authorization***], which is described later.

Deploying Web Service

Deploying the Materials

Materials for providing the Web Services are now completed in the steps above.
Then, the created modules are retrieved as user modules. War files are created and are deployed to Resin.

Follow the steps below to create the user modules by e Builder.

1. Right click on the project, and select [Export].
2. Select [e Builder]-[imm file], and click [Next].
3. Select any location as output folder, and click [Finish].
4. After a while, sample_provider-1.0.0.imm file is created in the output folder.

Follow the steps below to retrieve the imm file as a user module by e Builder.

1. Open [juggling.im] of the project which was used by e Builder when the environment was built.
2. Open the [User Module] tab, and click the [Add Module] icon on the upper right.
3. Select the sample_provider-1.0.0.imm file which was created by e Builder, and open it.

Note

If dependency relationships are insufficient, error message would be displayed in the upper part.
In such a case, please click the error message and add the missing modules.

4. Save juggling.im, create the war using the same steps as those at environment build, and deploy it to Resin.
5. Restart Resin.

Then, perform tenant environment setup using the steps below :

1. Open System Management Screen, and log in as a system administrator.
   http://<HOST>:<PORT>/<CONTEXT_PATH>/system/login
2. Click [Tenant Environment Setup].

Note

In case “Tenant environment is the latest one. There are no modules that need the setup.” is displayed, no further operations would be necessary.

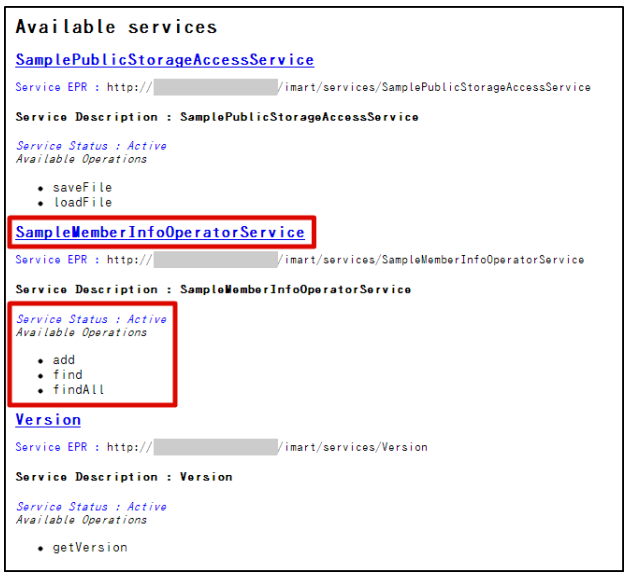
3. Then, click [Tenant Environment Setup].
4. Click [OK] in the confirmation message.

Confirming Deployment

When the Resin restart and tenant environment setup are completed, please check if the created Web service has been deployed.

Follow the steps below to confirm that the deployment has been performed successfully.

1. Access the URL below :
   http://<HOST>:<PORT>/<CONTEXT_PATH>/services/listServices
2. It is a success if it is confirmed that the character string of [Service Status : Active] and the names of each function are displayed in [SampleMemberInfoOperatorService].



Setting Access Authority by using Authorization

In the preceding descriptions of this tutorial, Resource URIs for authorization that are allocated to the Web Services are the ones accessible by any user regardless of the authentication status.

- service://intra-mart.jp/public-resources/welcome-to-intramart

In case you need to offer a Web service as a formal version, you need to register the authorized resource to allow detailed settings of access authority of the Web service.
In case of a Web service, [service] should be typically used as a scheme of [Resource URI], which is a key to the authorized resource.
For example, define the Resource URI as shown below :

- service://sample_provider/web_service/member_info_operator

In case you want to manage each function of a Web service by individually setting the authority, you should define the Resource URI for each function.
For example, if you want to make different authority settings for each of [add()], [find()], and [findAll()] which were published by the sample Web service, [Resource URI] is defined for creating each of the 3 resources.

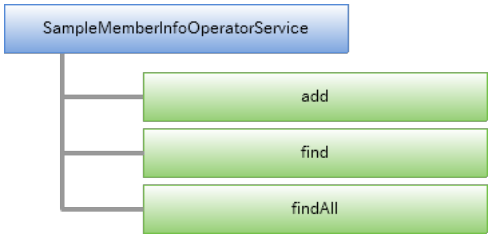
- service://sample_provider/web_service/member_info_operator/add
- service://sample_provider/web_service/member_info_operator/find
- service://sample_provider/web_service/member_info_operator/findAll

Creating the Authorized Resource

Once the [Resource ID] for authority setting is determined, please create the materials for tenant environment setup.
Required materials are as follows :

- Authorization Resource Setting File
- Setup Configuration File for Tenant Environment Setup

In this chapter the resource with the following configuration is registered in the authorization.



Note

Authorization resource can be registered from the [Authorization Setting Screen] of Tenant Management Function instead of the materials for tenant environment setup.
If there is a need to change the authorization resource frequently during the development of Web service, it would be convenient to make the settings from the Authorization Setting Screen.
For more information about the ways to operate Authorization Setting Screen, please refer to [Adding Resource] in [**Setting Authorization in Tenant Administrator Operations Guide**].

Each of the required materials should be created in the locations of the project below. Contents of the sample materials are as follows :

- Authorization Resource Setting File

src/main/storage/system/products/import/basic/sample_provider/sample_provider-authz-resource.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.intra-mart.jp/authz/imex/resource">

  <authz-resource id="sample_provider-service" uri="service://sample_provider/web_service/member_info_operator">
    <display-name>
      <name locale="ja">SampleMemberInfoOperatorService</name>
    </display-name>
    <parent-group id="web-services" />
  </authz-resource>

  <authz-resource uri="service://sample_provider/web_service/member_info_operator/add">
    <display-name>
      <name locale="ja">add</name>
    </display-name>
    <parent-group id="sample_provider-service" />
  </authz-resource>

  <authz-resource uri="service://sample_provider/web_service/member_info_operator/find">
    <display-name>
      <name locale="ja">find</name>
    </display-name>
    <parent-group id="sample_provider-service" />
  </authz-resource>

  <authz-resource uri="service://sample_provider/web_service/member_info_operator/findAll">
    <display-name>
      <name locale="ja">findAll</name>
    </display-name>
    <parent-group id="sample_provider-service" />
  </authz-resource>

</root>
```

Note

Authorization resource is created for the Web service itself and for the [Resource URI] allocated to each function.

Parent resource group [web-services] to register the authorization resources that are related to the Web service is provided with its initial state.
Therefore, the parent resource group of [SampleMemberInfoOperatorService] which is the top layer of sample Web service should be set to [web-services].

- Setup Configuration File for Tenant Environment Setup

src/main/conf/products/import/basic/sample_provider/import-sample_provider-config-1.xml

```
<import-data-config xmlns="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config import-data-config.xsd">

  <tenant-master>
    <authz-resource-file>products/import/basic/sample_provider/sample_provider-authz-resource.xml</authz-resource-file>
  </tenant-master>

</import-data-config>
```

Changing services.xml

After the creation of setup materials is completed, authority setting file (services.xml) to access each function of the Web service should be changed.

- src/main/webapp/WEB-INF/services/sample_member_info/META-INF/services.xml

Please rewrite the service://intra-mart.jp/public-resources/welcome-to-intramart part to the defined [Resource URI].
Source code of services.xml is as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceGroup>
  <service name="SampleMemberInfoOperatorService">
    <parameter name="ServiceClass">sample.web_service.provider.MemberInfoOperatorService</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>

    <parameter name="authz-uri">service://sample_provider/web_service/member_info_operator</parameter>

    <operation name="add">
      <parameter name="authz-uri">service://sample_provider/web_service/member_info_operator/add</parameter>
    </operation>

    <operation name="find">
      <parameter name="authz-uri">service://sample_provider/web_service/member_info_operator/find</parameter>
    </operation>

    <operation name="findAll">
      <parameter name="authz-uri">service://sample_provider/web_service/member_info_operator/findAll</parameter>
    </operation>
  </service>
</serviceGroup>
```

Note

For the setting contents of services.xml, please also refer to [**About services.xml in Web Services Authentication and Authorization Specifications**].

Redeploying the Materials

After the update of the Web service setting file is completed, please perform the redeployment.
Please follow the steps in [***Deploying the Materials***] and redeploy sample_provider-1.0.0.imm.

Warning

After the redeployment, please do not fail to do the tenant environment setup from the System Management Screen.

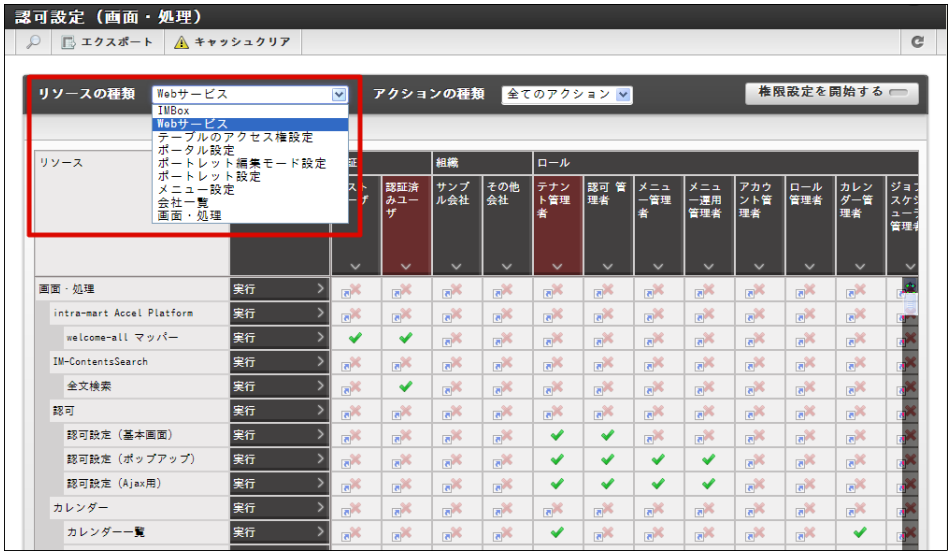
Setting Authority in Authorization

After the completion of the redeployment, open the Authorization Setting Screen, and actually set the access authority for the Web service.

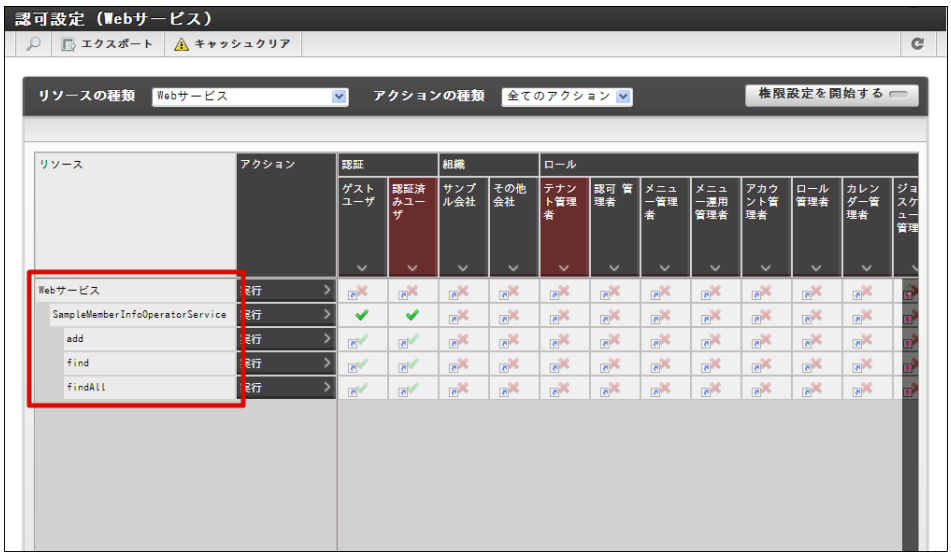
Please follow the steps below to set the access authority.

1. Open the log-in screen for the general users, and log in as a tenant administrator.
   http://<HOST>:<PORT>/<CONTEXT_PATH>/login
2. Open the site map, and click [Authorization] in [Tenant Management] category.

3. Select [Web Service] from [Resource Types].



4. Please check that [SampleMemberInfoOperatorService] is displayed in the grid of authorization setting and [add], [find], and [findAll] are displayed beneath it.



5. Please set authority to the condition of arbitrary taget user for [add], [find], and [findAll].

Note

For more information about the ways to set authorities, please refer to [**Setting Authorizations in Tenant Administrator Operations Guide**].

In order to check if the access authorities actually function as set, it is necessary to provide [Web Service Client].

Creating Web Service Client

Topics

Overview of Creation Steps

This chapter describes the necessary steps in Java to use operations published as Web services.
In this sample program a stub class is created from WSDL, and Web Service Client is implemented.
By utilizing the stub, you can invoke the Web service without worrying about XML.

Web service call using the stub can be performed in the following 3 steps :

1. Specify WSDL and generate the stub class.
2. Create the execution class which calls Web Service.
3. Access the Web service by using the execution class above.

In this tutorial necessary procedures until the Web service call described in [***Creating Web Service Provider***] will be presented.

Preparing the Development Environment

Installation of the development environment (e Builder, Resin, and intra-mart Accel Platform) should be made in a similar way to that in [***Preparing the Development Environment***] of Web Service Provider .
In this tutorial the steps for creating and developing the following project are described :

Group ID	mypackage (default setting is used)
Version	1.0.0 (default setting is used)
Project Name	sample_client

Once the installation of e Builder is completed, please create the project by [Modulle Project] following the steps below :
For more information please refer to [**Module Project Creation in e Builder Users Operations Guide**].

1. Click [File] - [New] - [Project].
2. Select [e Builder] - [Module Project] and click [Next].
3. Enter [sample_client] as a project name, and click [Finish].

After the completion of project creation, settings for the project should be made to make the API's of intra-mart Accel Platform available for use.
For more information, please refer to [**Basic functions of module development in e Builder Users Operations Guide**].

1. Right click on the project, and select [Property].
2. Select [e Builder] - [Module Assembly].
3. Select the folder with the same name as the context path generated by deploying the war in the Web archive directory.
4. Turn off all the check boxes in the automatic deployment list for the resource change.
5. Click [OK].

Resolving the Dependencies

After the completion of the project settings, dependency should be corrected.
In order to access Web Service Provider, it is necessary to depend on the [Web Service Authentication and Authorization Client] module.

Make the correction to the project dependency following the steps below :

1. Double click on module.xml which is placed in the root directory of the created project.
2. Open [Dependency] tab, and click [Add].
3. Enter the contents below, and click [OK].

ID	jp.co.intra_mart.im_ws_auth_client
Version	8.0.2

Note

You should basically specify the version number whichis supported.
If the API you need to use is included in the other version, please specify that version number.

4. After saving the module.xml file, open [module.xml] and delete the unnecessary tags (<tags>).
   Final source code would be as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<module conf:schemaLocation="urn:intramart:jackling:toolkit:configurations configurations.xsd"
        xmlns="urn:intramart:jackling:module" xmlns:conf="urn:intramart:jackling:toolkit:configurations"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemeLocation="urn:intramart:jackling:module module.xsd">

    <id>mypackage.sample_client</id>
    <version>1.0.0</version>
    <type>module</type>
    <name>${module.name}</name>
    <vendor>${module.vendor}</vendor>
    <description>${module.description}</description>

    <dependencies>
        <dependency>
            <module-id>jp.co.intra_mart.im_ws_auth_client</module-id>
            <verified-version min="8.0.2">8.0.2</verified-version>
        </dependency>
    </dependencies>
</module>
```

5. Open [Dependency Layer] tab. it would be a success if dependencies are resolved and various modules are displayed.

Creating Stub Class

Create the stub class. Stub class is created from WSDL by using the Web service tool.

These steps assume that the conditions below are met in the environment where these steps are performed.

- Apache Ant has been installed.
- [**Web Service Tool**] exists.
- WSDL published by Web Service Provider exists.

Hereafter the directory to which Web service tool is deployed is stated as %WEBSERVICE_TOOL_HOME%.

Note

If the build tool [Ant] has not been installed yet, please install it by referring to the site below :

Ja-Jakarta Project Ant Installation (Japanese)
Apache Ant Manual - Installing Apache Ant (English)

1. Various settings to generate the stub are made.

   1. Edit StubGen.properties.

      Edit %WEBSERVICE_TOOL_HOME%/StubGen.properties.
      Set the value of `wsdlfilename` to `http://<HOST>:<PORT>/<CONTEXT_PATH>/services/SampleMemberInfoOperatorService?wsdl`.
      `<HOST>`, `<PORT>`, and `<CONTEXT_PATH>` should be changed to the ones pointing to the application server on which Web Service Provider is running.

      Descriptions of eah module are provided below :

Property Name	Required	Description
wsdlfilename	O	Specify the file path or URL of WSDL.
destDir	O	Specify the directory which outputs the stub class.
imartDir	O	Specify the directory in which intra-mart Accel Platform is deployed. This setting is required to deploy the libraries necessary for the creation of the stub to the class path. Example) In case Resin exists at /resin and imart.war has already been deployed, /resin/webapps/imart is specified as the directory specfied by imartDir.

      Warning

      In the directory specified as `imartDir`, Web Service Authentication and Authorization module needs to have been installed.

      Warning

      In case the path is specified in Windows environment, please use / or \ as a delimiter.

   2. Specify the environment information to execute StubGen.

      Specify the directory in which Apache Ant is installed.
      Edit %WEBSERVICE_TOOL_HOME%/StubGen.bat (for StubGeb.sh for Unix type OS).
      Specify the directory in which Ant is installed for the environment variable [ANT_HOME].

      (for Windows type OS)

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

      (for Unix type OS)

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

2. Run StubGen.

   Execute the enclosed batch file.

   (for Windows type OS)

   ```
   > %WEBSERVICE_TOOL_HOME%\StubGen.bat
   ```

   (for Unix type OS)

   ```
   $ sh %WEBSERVICE_TOOL_HOME%/StubGen.sh
   ```

   Stub class information is generated under the directory specified by `destDir` of StubGen.properties.

Note

Please execute StubGen while WSDL specified by `wsdlfilename` can be obtained.

Note

Once StubGen is executed, the following stub class information is generated under the directory specified by `destDir`.

- stub.jar : jar file in which stub class file is enclosed
- under classes directory : stub class file
- under src directory : Stub Java file

Please use them as required.

Implementing Web Service Client

Implement Web Service Client.

Add stub class information created by [***Creating Stub Class***] to the class path.

1. Right click on e Builder project, and select [Build Path] - [Build Path Structure].
2. Add `stub.jar` which was generated under `destDir` by [***Creating Stub Class***].

Create Web Service execution class on e Builder following the procedures below.

1. Right click on the project and select [New] - [Class].
2. Enter the contents below and click [OK].

Package	sample.web_service.client
Name	MemberInfoOperatorRunner

3. MemberInfoOperatorRunner.java is created under `src/main/java` of project.
4. Please implement the following :

```
package sample.web_service.client;

import java.rmi.RemoteException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import jp.co.intra_mart.foundation.web_service.util.impl.WSAuthDigestGenerator4WSSE;

import org.apache.axis2.AxisFault;

import sample.web_service.provider.SampleMemberInfoOperatorServiceStub;
import sample.web_service.provider.SampleMemberInfoOperatorServiceStub.Add;
import sample.web_service.provider.SampleMemberInfoOperatorServiceStub.Find;
import sample.web_service.provider.SampleMemberInfoOperatorServiceStub.FindAll;
import sample.web_service.provider.SampleMemberInfoOperatorServiceStub.FindAllResponse;
import sample.web_service.provider.SampleMemberInfoOperatorServiceStub.FindResponse;
import sample.web_service.provider.SampleMemberInfoOperatorServiceStub.Member;
import sample.web_service.provider.SampleMemberInfoOperatorServiceStub.WSUserInfo;

/**
 * スタブを利用してSampleMemberInfoOperatorServiceのオペレーションを実行するクライアントクラスのサンプルです。
 */
public class MemberInfoOperatorRunner {

    // ホスト名、ポート番号、コンテキストパスは適宜置き換えてください。
    private static final String ENDPOINT = "http://localhost:8080/imart/services/SampleMemberInfoOperatorService";

    private static final String USER_CD = "aoyagi";

    private static final String PASSWORD = "aoyagi";

    // 実際にはプロバイダから提供された接続先ログイングループID/テナントIDを設定します。
    private static final String LOGIN_GROUP_ID = "default";

    public static void main(final String[] args) {
        try {
            new MemberInfoOperatorRunner().add();
        } catch (final RemoteException e) {
            e.printStackTrace();
        }
    }

    /**
     * SampleMemberInfoOperatorService#add を実行するサンプルです。<br>
     * 固定的にメンバー情報を追加します。
     */
    public void add() throws RemoteException {
        // Webサービス・オペレーションへのパラメータを作成します。
        final Add parameter = new Add();
        parameter.setWsUserInfo(generateWSUserInfo());

        final Member member = new Member();
        member.setId("test");
        member.setName("テストユーザ");
        member.setAge(30);
        member.setMarried(true);
        final Calendar cal = Calendar.getInstance();
        cal.set(Calendar.YEAR, 1982);
        cal.set(Calendar.MONTH, Calendar.JUNE);
        cal.set(Calendar.DAY_OF_MONTH, 12);
        member.setBirthDate(cal);
        parameter.setMember(member);

        // Webサービスを実行します。
        final SampleMemberInfoOperatorServiceStub client = getStub();
        client.add(parameter);

        // 実行結果を標準出力します。
        System.out.println("Success.");
    }
```

```
    }

    /**
     * SampleMemberInfoOperatorService#find を実行するサンプルです。<br>
     * ID "test" のメンバーを検索し、メンバー情報を標準出力します。
     */
    public void find() throws RemoteException {
        // Webサービス・オペレーションへのパラメータを作成します。
        final Find parameter = new Find();
        parameter.setWsUserInfo(generateWSUserInfo());
        parameter.setId("test");

        // Webサービスを実行します。
        final SampleMemberInfoOperatorServiceStub client = getStub();
        final FindResponse response = client.find(parameter);
        final Member member = response.get_return();

        // 実行結果を標準出力します。
        printMember(member);
    }

    /**
     * SampleMemberInfoOperatorService#findAll を実行するサンプルです。<br>
     * 現在のメンバー情報の数を標準出力します。
     */
    public void findAll() throws RemoteException {
        // Webサービス・オペレーションへのパラメータを作成します。
        final FindAll parameter = new FindAll();
        parameter.setWsUserInfo(generateWSUserInfo());

        // Webサービスを実行します。
        final SampleMemberInfoOperatorServiceStub client = getStub();
        final FindAllResponse response = client.findAll(parameter);
        final Member[] members = response.get_return();

        // 実行結果を標準出力します。
        if (members == null) {
            System.out.println("Member count : 0");
        } else {
            System.out.println("Member count : " + members.length);
        }
    }

    private WSUserInfo generateWSUserInfo() {
        final WSUserInfo info = new WSUserInfo();
        info.setLoginGroupID(LOGIN_GROUP_ID);
        info.setUserID(USER_CD);
        info.setPassword(WSAuthDigestGenerator4WSSE.createWsseAuthString(USER_CD, PASSWORD));
        info.setAuthType(WSAuthDigestGenerator4WSSE.authType);

        return info;
    }

    private SampleMemberInfoOperatorServiceStub getStub() throws AxisFault {
        final SampleMemberInfoOperatorServiceStub client = new SampleMemberInfoOperatorServiceStub(ENDPOINT);

        return client;
    }

    private void printMember(final Member member) {
        System.out.println("id       : " + member.getId());
        System.out.println("name     : " + member.getName());
        System.out.println("age      : " + member.getAge());
        System.out.println("married  : " + member.getMarried());
        final SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
        System.out.println("birthday : " + formatter.format(member.getBirthDate().getTime()));
    }

}
```

Note

User information for authentication/authorization is set as parameters of Web services.
In this sample the authentication type [WSSE] is used.
For more information about authentication type [WSSE], please refer to [**Authentication/Authorization in Web Service Authentication and Authorization Specifications**].

In this sample the password digest is created by using WSAuthDigestGenerator4WSSE class.
In the authentication type [WSSE], UsernameToken format of WS-Security is adopted as a method to digest passwords.
WSAuthDigestGenerator4WSSE class is an utility specialized for the creation of its password digest.
Password digest is created based on the [User Code] and [Password].
For the details about WSAuthDigestGenerator4WSSE, please refer to [**WSAuthDigestGenerator4WSSE API List**].

Note

Information which is set to wsUserInfo should be compatible with the settings on Web service provider side.
In the sample program, wsUserInfo information is defined inside the [generateWSUserInfo] method.

Note

If SOAPFault error is thrown by Web Service Provider, AxisFault will be thrown as an exception when the stub is executed.
For the detaiils about AxisFault, please refer to [**Axis2 API Document**].

It should also be noted that on the Web Service Provider side authentication and authorization are conducted based on the authentication information (WSUserInfo) which was set when the Web service was called.
If user information is invalid for such reasons as non existing user or incorrect password, or if the required authority to execute the Web service does not exist, AxisFault will be thrown as an exception.
Error code for the detected problem can be obtained by the `getFaultCode` method of AxisFault.
For the details about these codes, please refer to [**Authentication/Authorization SOAP Fault Code in Web Service Authentication and Authorization Specifications**].

Accessing Web Service

Access Web Service as a Java application on e Builder.

- Please register member information following the steps below :

  1. In the project, right click on `src/main/java` – `sample.web_service.client` – `MemberInfoOperatorRunner.java` , and select [Run] - [Java Application].
  2. It is successful if the following message is displayed on the console.

```
Success.
```

Note

In case Unsupported major.minor version error occurs, please check if the version of Java compiler and that of jre used for execution do match.

- Search the member information following the steps below :

  1. Change the method that is executed by the `main` method of java class created by [***Implementing Web Service Client***] from `add` to `find`.
  2. In the project, please right click on `src/main/java` – `sample.web_service.client` – `MemberInfoOperatorRunner.java`, and select [Run] - [Java Application].
  3. It is successful if the following message is displayed on the console.

```
id      : test
name    : Test User
age     : 30
barried : true
birthday : 1982-06-12
```

- Number of items of member information can be displayed following the steps below :

  1. Change the method that is executed by the `main` method of java class created by [***Implementing Web Service Client***] from `find` to `findAll`.
  2. In the project, please right click on `src/main/java` – `sample.web_service.client` – `MemberInfoOperatorRunner.java`, and select [Run] - [Java Application].
  3. It is successful if the following message is displayed on the console.

```
Member count : 1
```

Note

Number of items of member information that have been added by [add] are displayed.

Appendix

Topics

Troubleshooting

In case you use WSDL provided by https

If the WSDL below is provided by https, it is necessary to obtain and register the server certificate of connection destination to execute the Web service by the stub.

- https://<HOST>/<CONTEXT_PATH>/services/SampleWebService?wsdl

Please obtain and register the server certificate following the steps below :

1. Obtain the server certificate of connection destination.
   There are several ways to obtain server certificates. Here the method of obtaining the certificate using Internet Explorer 9 in Windows environment will be shown.

   1. Open Internet Explorer 9, and enter the URL of WSDL for access.
   2. Press the Alt key to open the menu bar, and select [Tool] - [Internet Option].
   3. Open the [Contents] tab, and click [Certificate].
   4. Select the certificate you want to obtain, and click [Export].
   5. Please proceed with the wizard, and save the server certificate.
2. By using the keytool included in JDK, add the server certificate to the keystore.
   Example : The server certificate file is stored at C:\temp\server.crt, and is added to the keystore entry with an alias “sample_alias”.

```
keytool -import -alias sample_alias -file C:\temp\server.crt
```

Note

If you run the command above, keystore is created in the [.keystore] file in the user's home directory.
For more information about keytool, please refer to “Keytool for JDK document - Management Tool for Keys and Certificates”.

http://docs.oracle.com/javase/jp/7/technotes/tools/windows/keytool.html (Japanese)
http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html (English)

3. Please add “javax.net.ssl.trustStore” to the system property of JavaVM in the application server.
   Example : Resin is to be installed to C:\resin, and user name is user_name.

C:\resin\conf\resin.properties

```
jvm_args : -Djavax.net.ssl.trustStore="C:\Users\user_name\.keystore"
```

Note

In case jvm_args already exists, please put one single-byte space character at the end before adding it.

Even if the URL of WSDL starts with https, end point should be specified explicitly when the stub is used if the end point stated in WSDL is not https.

```
// Specify the end point as the first argument of stab constructor.
new SampleWebServiceStub("https://localhost/imart/services/SampleWebService");
```

In case “Specified request has failed” occurs

In case this symptom occurs, probable cause of it would be as follows:

- You may not have the authority to execute the specified Web service.

For more information about the resolution of this problem, please refer to [wsse:RequestFailed] in [**Authentication/Authorization SOAP Fault Code in Web Service Authentication and Authorization Specifications**].

In case “Specified RequestSecurityToken could not be understood” occurs

In case this symptom occurs, probable cause of it would be as follows:

- Authentication module which supports the authentication type may not exist.

For more information about the resolution of this problem, please refer to [wsse:BadRequest] in [**Authentication/Authorization SOAP Fault Code in Web Service Authentication and Authorization Specifications**].

In case “Request is invalid or in wrong format” occurs

In case this symptom occurs, probable cause of it would be as follows:

- User information may not exist in the SOAP body.
- Name of the element which stores the user information may not be [wsUserInfo].
- Compilation method of Java class to be published as a Web service may be wrong.

Please open the URL of WSDL by the browser, and check the argument name in the Web service function definition.

[*] Correct

```
<xs:element name="add">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="wsUserInfo" type="ax22:WSUserInfo" nillable="true" minOccurs="0"/>
      <xs:element name="member" type="ax24:Member" nillable="true" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

[*] Incorrect

```
<xs:element name="add">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="param0" type="ax22:WSUserInfo" nillable="true" minOccurs="0"/>
      <xs:element name="param1" type="ax24:Member" nillable="true" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

If the argument names are [param0] and [param1] as shown in the code above, your compilation method of Java class is wrong.

For more information about the resolution of this problem, please refer to [wsse:InvalidRequest] in [**Authentication/Authorization SOAP Fault Code in Web Service Authentication and Authorization Specifications**].

Sample Code

Specifying the timeout value for Web Service

In order to specify the timeout value for Web Service execution, timeout value should be specified to the stub.

```
final SampleWebServiceStub client = new SampleWebServiceStub(ENDPOINT);
// Timeout value is set to 5 seconds (5000 milliseconds)
client._getServiceClient().getOptions().setTimeOutInMilliSeconds(5000);
```

Sample for Sending/Receiving Binary Files

By specifying “byte[]” to the type of argument and return value of the method of Web Service Provider implementation class which is published as Web service, you can send and receive binary files.
Byte arrays that are passed are automatically encoded in Base64, and are sent or received as SOAP messages.
Steps to activate the sample program which sends and receives binary files are described below.

Creating Web Service Provider

Create Web Service Provider in a similar way as [***Creating Web Service Provider***].
In this sample [***Preparing the Development Environment***] and [***Resolving Dependencies***] are eliminated.

Please make ``sample.web_service.provider.PublicStorageAccessService.java `` available for use.
Methods [loadFile()] and [saveFile()] which are defined in this class are published as Web services.
Contents of PublicStorageAccessService.java are as follows. Please refer to [***Implementing the Web Service Processes***] to find out how to create the class.

```
package sample.web_service.provider;

import java.io.IOException;

import jp.co.intra_mart.foundation.service.client.file.PublicStorage;
import jp.co.intra_mart.foundation.web_service.auth.WSUserInfo;

import org.apache.axis2.AxisFault;

public class PublicStorageAccessService {

   public byte[] loadFile(final WSUserInfo wsUserInfo, final String path) throws AxisFault {
      final PublicStorage storege = new PublicStorage(path);
      try {
         return storege.load();
      } catch (final IOException e) {
         throw AxisFault.makeFault(e);
      }
   }

   public Boolean saveFile(final WSUserInfo wsUserInfo, final String path, final byte[] data) throws AxisFault {
      final PublicStorage storege = new PublicStorage(path);
      try {
         storege.save(data);
         return Boolean.TRUE;
      } catch (final IOException e) {
         throw AxisFault.makeFault(e);
      }
   }

}
```

Create services.xml to publish PublicStorageAccessService as Web Service.
Place services.xml to `src/main/webapp/WEB-INF/services/sample_public_storage/META-INF/services.xml` .
Contents of services.xml are as shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceGroup>
   <service name="SamplePublicStorageAccessService">
      <parameter name="ServiceClass">sample.web_service.provider.PublicStorageAccessService</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>

      <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>

      <operation name="loadFile">
         <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
      </operation>

      <operation name="saveFile">
         <parameter name="authz-uri">service://intra-mart.jp/public-resources/welcome-to-intramart</parameter>
      </operation>

   </service>
</serviceGroup>
```

Deploy it in a similar way as [***Deploying Web Service***].

Creating Web Service Client

Implement Web Service Client in a similar way as [***Creating Web Service Client***].
In this sample [***Preparing the Development Environment***] is eliminated.

Create the stub to access `SamplePublicStorageAccessService` .
Stub class should be created as described in [***Creating Stub Class***].
URL of WSDL is `http://<HOST>:<PORT>/<CONTEXT_PATH>/services/SamplePublicStorageAccessService?wsdl` .
`<HOST>`, `<PORT>`, and `<CONTEXT_PATH>` should be changed to point to the application server on which Web Service Provider is running.

Add the stub class information created as above to the class path in a similar way to the steps in [***Implementing Web Service Client***].

Create Web Service execution class `sample.web_service.client.PublicStorageAccessRunner.java` .
Contents of PublicStorageAccessRunner.java are as follows.

```
package sample.web_service.client;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.rmi.RemoteException;
```

```
import javax.activation.DataHandler;

import jp.co.intra_mart.foundation.web_service.util.impl.WSAuthDigestGenerator4WSSE;

import org.apache.axis2.AxisFault;

import sample.web_service.provider.SamplePublicStorageAccessServiceStub;
import sample.web_service.provider.SamplePublicStorageAccessServiceStub.LoadFile;
import sample.web_service.provider.SamplePublicStorageAccessServiceStub.LoadFileResponse;
import sample.web_service.provider.SamplePublicStorageAccessServiceStub.SaveFile;
import sample.web_service.provider.SamplePublicStorageAccessServiceStub.WSUserInfo;

/**
 * スタブを利用してSamplePublicStorageAccessServiceのオペレーションを実行するクライアントクラスのサンプルです。
 */
public class PublicStorageAccessRunner {

    // ホスト名、ポート番号、コンテキストパスは適宜置き換えてください。
    private static final String ENDPOINT = "http://localhost:8080/imart/services/SamplePublicStorageAccessService";

    private static final String USER_CD = "aoyagi";

    private static final String PASSWORD = "aoyagi";

    private static final String LOGIN_GROUP_ID = "default";

    public static void main(final String[] args) {
        try {
            new PublicStorageAccessRunner().loadFile();
        } catch (final IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * SamplePublicStorageAccessService#loadFile を実行するサンプルです。<br>
     * パブリックストレージの"sample/test.data"をカレントディレクトリの"temp.data"にコピーします。
     */
    public void loadFile() throws IOException {
        // Webサービス・オペレーションへのパラメータを作成します。
        final LoadFile parameter = new LoadFile();
        parameter.setWsUserInfo(generateWSUserInfo());
        parameter.setPath("sample/test.data");

        // Webサービスを実行します。
        final SamplePublicStorageAccessServiceStub client = getStub();
        final LoadFileResponse response = client.loadFile(parameter);
        final DataHandler handler = response.get_return();

        // カレントディレクトリのファイル"temp.data"に内容を保存します。
        final File file = new File("temp.data");
        final FileOutputStream fos = new FileOutputStream(file);
        final InputStream in = handler.getInputStream();

        final byte[] buff = new byte[1024];
        try {
            while (true) {
                final int len = in.read(buff);
                if (len != -1) {
                    fos.write(buff, 0, len);
                } else {
                    break;
                }
            }
            fos.flush();
        } finally {
            fos.close();
            in.close();
        }

        // 実行結果を標準出力します。
        System.out.println("Out file : " + file.getAbsolutePath());
    }

    /**
     * SamplePublicStorageAccessService#saveFile を実行するサンプルです。<br>
     * カレントディレクトリの"temp.data"をパブリックストレージの"sample/test.data"にコピーします。
     */
    public void saveFile() throws RemoteException, MalformedURLException {
        // Webサービス・オペレーションへのパラメータを作成します。
        final SaveFile parameter = new SaveFile();
        parameter.setWsUserInfo(generateWSUserInfo());
        parameter.setPath("sample/test.data");
        final File file = new File("temp.data");
        parameter.setData(new DataHandler(file.toURI().toURL()));
```

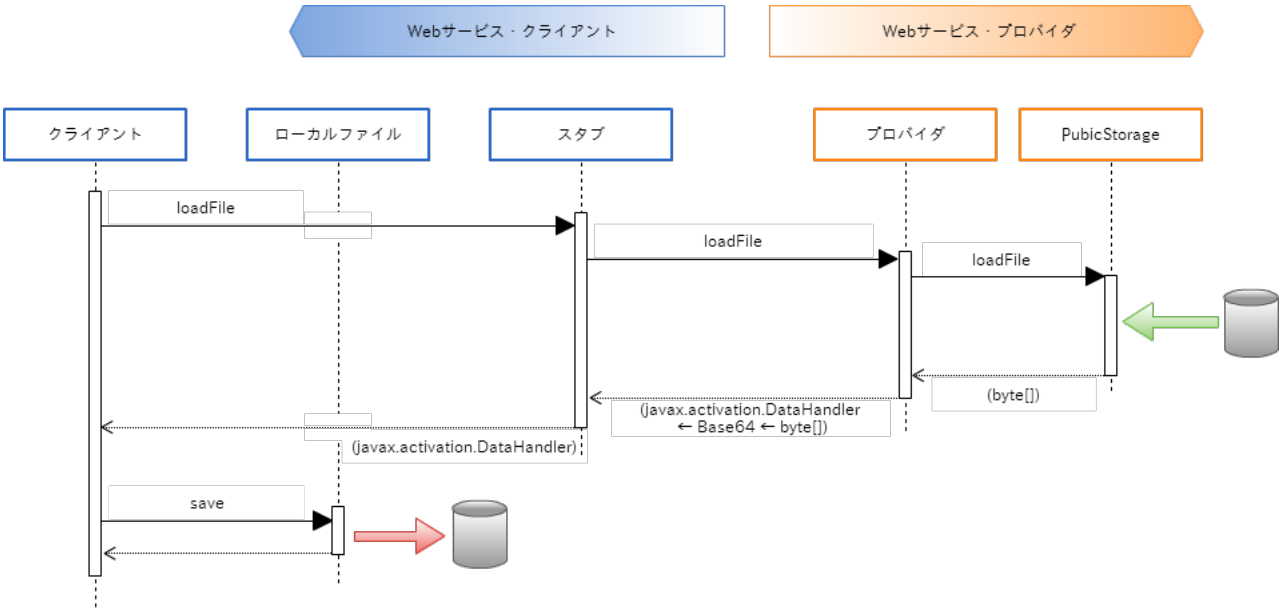
```
        // Webサービスを実行します。
        final SamplePublicStorageAccessServiceStub client = getStub();
        client.saveFile(parameter);

        // 実行結果を標準出力します。
        System.out.println("Success.");
    }

    private WSUserInfo generateWSUserInfo() {
        final WSUserInfo info = new WSUserInfo();
        info.setLoginGroupID(LOGIN_GROUP_ID);
        info.setUserID(USER_CD);
        info.setPassword(WSAuthDigestGenerator4WSSE.createWsseAuthString(USER_CD, PASSWORD));
        info.setAuthType(WSAuthDigestGenerator4WSSE.authType);

        return info;
    }

    private SamplePublicStorageAccessServiceStub getStub() throws AxisFault {
        final SamplePublicStorageAccessServiceStub client = new SamplePublicStorageAccessServiceStub(ENDPOINT);

        return client;
    }

}
```

Executing Web Service

This sample program sends and receives the contents of local files in Web Service Client and files in the PublicStorage on Web Service Provider side.

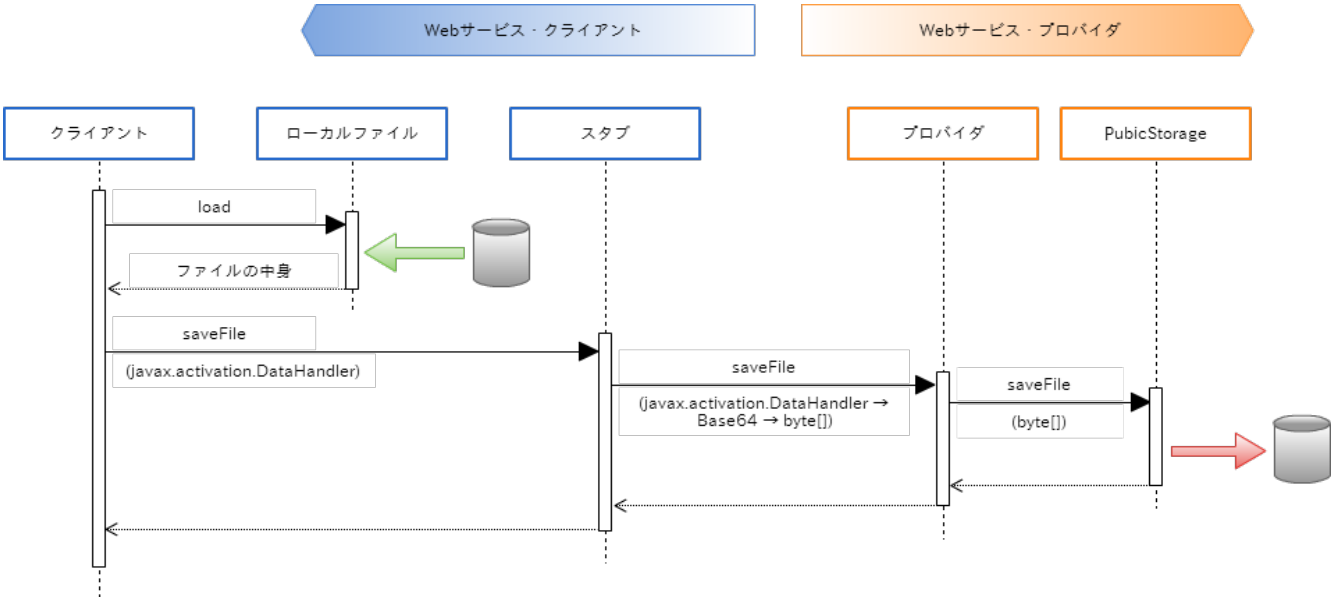
- Receive Binary Files (SamplePublicStorageAccessService#loadFile)

  Retrieve file contents from PublicStorage on Web Service Provider side through the Web service, and save them to the local files in Web Service Client.
  On e Builder environment, please right click on `PublicStorageAccessRunner.java` , and select [Run] - [Java Application].
  Process flow when `loadFile` is executed is as follows :



- Send Binary Files (SamplePublicStorageAccessService#saveFile)

  Load local files in Web Service Client and save them to PublicStorage on Web Service Provider side through the Web service.
  Change the method to be executed by `main` method of PublicStorageAccessRunner.java from `loadFile` to `saveFile` .
  On e Builder environment, please right click on `PublicStorageAccessRunner.java` , and select [Run] - [Java Application].
  Process flow when `saveFile` is executed is as follows:



Warning

If JavaBean is specified in the argument of the function to be published as a Web service, byte array type (byte[]) property in the JavaBean cannot normally send or receive data.
This limitation comes from the current specifications of Apache Axis2.
In case you send or receive binary files, please specify byte array (byte[]) as an argument of the function to be published as a Web service and not as a JavaBean property.