SOAP AIM Ingest Interface Specification

SOAP AIM Ingest Interface Specification 0.3.doc

Version 0.1

May 14, 2007

Cisco Systems, Inc

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Cisco Systems, Inc.	Initial revision	05/14/2007

Contents

1 . Introduction

1.1 Overview

This document defines the Cisco Systems, Inc. SOAP(Service Oriented Architecture Protocol) AIM(AVS Ingest Manager) ingest interface supported by the Cisco Content Delivery System (CDS).

This interface complies with SOAP 1.1 and 1.2 as defined by the WC3 specification. Which can be found at http://www.w3.org/TR/soap/.

The document is designed for developers wanting to integrate with the CDS, specifically the ingest of content.

1.2 Acronyms Defined

Acronym	Definition
CDS	Content Delivery System
AIM	AVS Ingest Manager
SOAP	Service Oriented Architecture Protocol
GSOAP	Generator Tools for Coding SOAP/XML Web Services in C and C++
WC3	World Wide Web Consortium
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
XSD	XML Schema Definition
WSDL	Web Services Description Language
XML	Extensible Markup Language
TNS	Target Name Space
TBD	To Be Defined
VOD	Video On Demand
SVOD	Subscriber Video On Demand

Table 1

1.3 Related Documents

[1] - RFC 2616 "Hypertext Transfer Protocol -- HTTP/1.1 | 1.2"
[2] - WC3 Recomendation "SOAP 1.2/1.1"
[3] - WC3 Recomendation "HTML 4.01"
[4] - WC3 Recomendation "XML"
[5] - WC3 Recomendation "XSD"
[6] - WC3 Recomendation "WSDL"
[7] - gsoap 2.7.9 http://www.cs.fsu.edu/~engelen/soap.html

2 . Configuration

2.1 Configuration

**To configure AIM to use the SOAP interface requires a database update, which is
possible through the CDSM. If the CDSM is not available the AIM default is to load the
ingest SOAP and ingest CORBA(ISA) interfaces. If you have a tool to update the AIM config, the the tag is 'INGEST_INTERFACE_MASK' the value is 'W,X,Y,Z', 4 numeric values seperated with a coma. The following are the valid values**

```
/*****************************************************************/
/* syntax:  W,X,Y,Z                                              */
/* 'W'=Ingest      'X'=Encrypt      'Y'=Store      'Z'=BackOffice  */
/* 0- disabled     0- disabled     0- disabled   0- disabled       */
/* 1- ISA          1- Verimatrix   1- ISA        1- TotalManage    */
/* 2- SOAP Cisco   2- WidVine      2- FSI                          */
/* 4- SOAP Prodis                  4- NGOD                         */
/*                                                               */
/*                                                               */
/* NOTE: W (Ingest) interface can be cumulative (i.e. 3 turns on   */
/* ISA and the SOAP cisco interfaces                             */
/*                                                               */
/*                                                               */
/*****************************************************************/
```

2.2 Connections

If the client is using SOAP though HTTP 1.1, then the HTTP 1.1 connection persistance applies. If using HTTP 1.0, a new connection is created for each request.

3 . SOAP Exports (callable)

3.1 IngestPackage

Signature

```
SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__IngestPackage(
  struct soap *pSOAP,
  struct _CISCOAIM__IngestPackage *pIn,
  struct _CISCOAIM__IngestPackageResponse *pOut)
```

Structs

```
struct _CISCOAIM__IngestPackage
{
    char *ADIURL;
    char *PackageName;  // Optional element, if not provided uses name in adi
    int   MetaDataOnly; // Optional element, defaults to 0=NO
    int   DoAsync;      // Optional element, defaults to 0=NO
};

struct _CISCOAIM__IngestPackageResponse
{
    char *IngestResult;
};
```

Parameter Description

```
REQUEST:
'ADIURL' - the location of the package XML. ie. 'ftp://Im:Pass@my.box.info/here/ADI.XML'
'PackageName' - Identification of the package.
'MetaDataOnly'- [0-NO|1-YES] Only add the XML metadata. Useful if the content is already
                             on the store location.
'DoAsync'     - [0-NO|1-YES] Imediately return from the IngestPackage call, do not wait
                             until complete. This is useful for clients that are single
                             threaded and process one package at a time.  Setting to '1'
                             will cause AIM to callback the application using the
                             imported AIMPackageNotification interface if available.  If
                             AIMPackageNotification is not avail, client will have to
                             use GetPackageStatus to know when complete. More details
                             are discussed in the AIMPackageNotification section.
RESPONSE:
'IngestResult' - did the operation complete successfully? If not......
```

3.2 DeletePackage

Signature

```
SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__DeletePackage(
  struct soap *pSOAP,
  struct _CISCOAIM__DeletePackage *pIn,
  struct _CISCOAIM__DeletePackageResponse *pOut)
```

Structs

```
struct _CISCOAIM__DeletePackage
{
       char *PackageName;
       int   MetaDataOnly; // Optional, default 0=NO
};

struct _CISCOAIM__DeletePackageResponse
{
    char *DeleteResult;
};
```

Parameter Description

REQUEST:

```
'PackageName'  - Identification for package to delete.
'MetaDataOnly' - Delete only the metadata? ie. 0=NO (deletes content), 1=YES (doesn't
                 delete content)
```

RESPONSE:

```
'DeleteResult' - did the operation complete successfully? If not......
```

3.3 UpdatePackage

UpdatePackage has almost the same behavior as IngestPackage. The exception being that the PackageName must exist. If it does the ADI.XML determines the updated information. So if content has been added to the package, only the new content and metadata will be added. Likewise if the content is no longer a part of the package, only the removed content will be deleted, along with the metadata.

Signature

```
SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__UpdatePackage(
  struct soap *pSOAP,
  struct _CISCOAIM__UpdatePackage *pIn,
  struct _CISCOAIM__UpdatePackageResponse *pOut)
```

Structs

```
struct _CISCOAIM__UpdatePackage
{
    char *ADIURL;
    char *PackageName;
    int   MetaDataOnly;
    int   DoAsync;
};

struct _CISCOAIM__UpdatePackageResponse
{
    char *UpdateResult;
}
```

Parameter Description

```
REQUEST:
'ADIURL' - the location of the package XML. ie. 'ftp://Im:Pass@my.box.info/here/ADI.XML'
'PackageName' - Identification of the package.
'MetaDataOnly'- [0-NO|1-YES] Only update the XML metadata. Useful if the content is
                             already on the store location.
'DoAsync'     - [0-NO|1-YES] Imediately return from the UpdatePackage call, do not wait
                             until complete. This is useful for clients that are single
                             threaded and process one package at a time.  Setting to '1'
                             will cause AIM to callback the application using the
                             imported AIMPackageNotification interface if available.  If
                             AIMPackageNotification is not avail, client will have to
                             use GetPackageStatus to know when complete. More details
                             are discussed in the AIMPackageNotification section.
RESPONSE:
'UpdateResult' - did the operation complete successfully? If not......
```

3.4 GetPackageStatus

Signature

```
SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__GetPackageStatus(
  struct soap *pSOAP,
  struct _CISCOAIM__GetPackageStatus *pIn,
  struct _CISCOAIM__GetPackageStatusResponse *pOut)
```

Structs

```
struct _CISCOAIM__GetPackageStatus
{
        char *PackageName;
};

struct _CISCOAIM__GetPackageStatusResponse
{
    char *StatusResult;
};
```

Parameter Description

```
REQUEST:
'PackageName'  - Identification for package to delete.

RESPONSE:
'StatusResult' - Will contain one of the following(currently minimal additional info)
                 COMPLETE - the package has been persisted in the system
                 FAILED   - error occured, with more information
                 PENDING  - waiting for available resources
                 RECOVERING - AIM is in the process of recovering the resource
                 INCOMPLETE - not completely ingested, with rough estimate in seconds
```

3.5 GetAllPackages

Signature

```
SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__GetAllPackages(
  struct soap *pSOAP,
  char *pLocation,
  struct _CISCOAIM__GetAllPackagesResponse *pOut)
```

Structs

```
struct _CISCOAIM__GetAllPackagesResponse
{
    struct CISCOAIM__List *PackageList;
};
```

Parameter Description

REQUEST:
No parameters

RESPONSE:

```
'PackageList' - Depends on what your typemap does for lists. The root implementation
                defines it as
```

```
struct CISCOAIM__List
{
    int  __sizestring;
    char **string;      // Null pointer, when list is size 0
};
```

4 . SOAP imports (optional)

This interface is OPTIONAL. If when the async flag is set, callbacks will be sent back to the client making the AIMPackageNotification SOAP call. The expected client response is 0 (which means ok you got it). The response can contain any value the client wants --- AIM does not use the result (but it does get logged with DEBUG logging -- which is a value of 2 in the AIM config table for AIM_DEBUG), but during testing you could use the value to sync specific external events.

4.1 AIMPackageNotification

Signature

```
SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__GetAllPackages(
  struct soap *pSOAP,
  char *pLocation,
  struct _CISCOAIM__GetAllPackagesResponse *pOut)
```

Structs

```
struct _IMPORT__AIMPackageNotification
{
        char *ADIURL;
        char *PackageName;
        char *Result
};

struct _IMPORT__AIMPackageNotificationResponse
{
        int NotificationResult;
};
```

Parameter Description

```
REQUEST:
'ADIURL' - the location of the package XML. ie. 'ftp://Im:Pass@my.box.info/here/ADI.XML'
'PackageName' - Identification of the package.
'Result'      - Same format as the other results(ie. IngestPackageResult,etc.)

RESPONSE:
'NotificationResult' - AIM accepts any valid 32bit signed integer.
```

5 . WSDL definitions

5.1 'CiscoAIM.wsdl'

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://cisco.aim.ns/CiscoAIM"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://cisco.aim.ns/CiscoAIM"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://cisco.aim.ns/CiscoAIM">
      <s:element name="IngestPackage">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="ADIURL" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="PackageName" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="MetaDataOnly" type="s:int"/>
            <s:element minOccurs="0" maxOccurs="1" name="DoAsync" type="s:int"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="IngestPackageResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="IngestResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="DeletePackage">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="PackageName" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="MetaDataOnly" type="s:int"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="DeletePackageResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="DeleteResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="UpdatePackage">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="ADIURL" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="PackageName" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="MetaDataOnly" type="s:int"/>
            <s:element minOccurs="0" maxOccurs="1" name="DoAsync" type="s:int"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="UpdatePackageResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="UpdateResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetPackageStatus">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="PackageName" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetPackageStatusResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="StatusResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetAllPackages">
      </s:element>
      <s:element name="GetAllPackagesResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="PackageList" type="tns:List"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="List">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded" name="string" nillable="true" type="s:string"/>
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="IngestPackageSoapIn">
    <wsdl:part name="parameters" element="tns:IngestPackage"/>
  </wsdl:message>
  <wsdl:message name="IngestPackageSoapOut">
    <wsdl:part name="parameters" element="tns:IngestPackageResponse"/>
  </wsdl:message>
  <wsdl:message name="DeletePackageSoapIn">
    <wsdl:part name="parameters" element="tns:DeletePackage"/>
  </wsdl:message>
  <wsdl:message name="DeletePackageSoapOut">
    <wsdl:part name="parameters" element="tns:DeletePackageResponse"/>
  </wsdl:message>
  <wsdl:message name="UpdatePackageSoapIn">
    <wsdl:part name="parameters" element="tns:UpdatePackage"/>
  </wsdl:message>
```

```
<wsdl:message name="UpdatePackageSoapOut">
  <wsdl:part name="parameters" element="tns:UpdatePackageResponse"/>
</wsdl:message>
<wsdl:message name="GetPackageStatusSoapIn">
  <wsdl:part name="parameters" element="tns:GetPackageStatus"/>
</wsdl:message>
<wsdl:message name="GetPackageStatusSoapOut">
  <wsdl:part name="parameters" element="tns:GetPackageStatusResponse"/>
</wsdl:message>
<wsdl:message name="GetAllPackagesSoapIn">
  <wsdl:part name="parameters" element="tns:GetAllPackages"/>
</wsdl:message>
<wsdl:message name="GetAllPackagesSoapOut">
  <wsdl:part name="parameters" element="tns:GetAllPackagesResponse"/>
</wsdl:message>
<wsdl:portType name="CiscoAIMSoap11">
  <wsdl:operation name="IngestPackage">
    <wsdl:input message="tns:IngestPackageSoapIn"/>
    <wsdl:output message="tns:IngestPackageSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="UpdatePackage">
    <wsdl:input message="tns:UpdatePackageSoapIn"/>
    <wsdl:output message="tns:UpdatePackageSoapOut"/>
 </wsdl:operation>
  <wsdl:operation name="DeletePackage">
    <wsdl:input message="tns:DeletePackageSoapIn"/>
    <wsdl:output message="tns:DeletePackageSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="GetPackageStatus">
    <wsdl:input message="tns:GetPackageStatusSoapIn"/>
    <wsdl:output message="tns:GetPackageStatusSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="GetAllPackages">
    <wsdl:input message="tns:GetAllPackagesSoapIn"/>
    <wsdl:output message="tns:GetAllPackagesSoapOut"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CiscoAIMSoap11" type="tns:CiscoAIMSoap11">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="IngestPackage">
    <soap:operation soapAction="CISCOAIM#IngestPackage" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="UpdatePackage">
    <soap:operation soapAction="CISCOAIM#UpdatePackage" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
```

```
    <wsdl:operation name="DeletePackage">
      <soap:operation soapAction="CISCOAIM#DeletePackage" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetPackageStatus">
      <soap:operation soapAction="CISCOAIM#GetPackageStatus" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetAllPackages">
      <soap:operation soapAction="CISCOAIM#GetAllPackages" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="CiscoAIMSoap12" type="tns:CiscoAIMSoap11">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="IngestPackage">
      <soap:operation soapAction="CISCOAIM#IngestPackage" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="UpdatePackage">
      <soap:operation soapAction="CISCOAIM#UpdatePackage" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DeletePackage">
      <soap:operation soapAction="CISCOAIM#DeletePackage" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetPackageStatus">
      <soap:operation soapAction="CISCOAIM#GetPackageStatus" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetAllPackages">
      <soap:operation soapAction="CISCOAIM#GetAllPackages" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CiscoAIM">
    <wsdl:port name="CiscoAIMSoap11" binding="tns:CiscoAIMSoap11">
      <soap:address location="http://localhost:8792/CiscoAIM"/>
    </wsdl:port>
    <wsdl:port name="CiscoAIMSoap12" binding="tns:CiscoAIMSoap11">
      <soap:address location="http://localhost:8793/CiscoAIM"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

5.2 'CiscoAIMNotification.wsdl'

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://cisco.aim.ns"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://cisco.aim.ns"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://cisco.aim.ns">
      <s:element name="AIMPackageNotification">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="ADIURL" type="s:string"/>
            <s:element minOccurs="1" maxOccurs="1" name="PackageName" type="s:string"/>
            <s:element minOccurs="1" maxOccurs="1" name="Result" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="AIMPackageNotificationResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="NotificationResult" type="s:int"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="AIMPackageNotificationSoapIn">
    <wsdl:part name="parameters" element="tns:AIMPackageNotification"/>
  </wsdl:message>
  <wsdl:message name="AIMPackageNotificationSoapOut">
    <wsdl:part name="parameters" element="tns:AIMPackageNotificationResponse"/>
  </wsdl:message>
  <wsdl:portType name="CiscoAIMNotificationSoap11">
    <wsdl:operation name="AIMPackageNotification">
      <wsdl:input message="tns:AIMPackageNotificationSoapIn"/>
      <wsdl:output message="tns:AIMPackageNotificationSoapOut"/>
    </wsdl:operation>
   </wsdl:portType>
  <wsdl:binding name="CiscoAIMNotificationSoap11" type="tns:CiscoAIMNotificationSoap11">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="AIMPackageNotification">
      <soap:operation soapAction="http://cisco.aim.ns/AIMPackageNotification" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
   </wsdl:binding>
  <wsdl:binding name="CiscoAIMNotificationSoap12" type="tns:CiscoAIMNotificationSoap11">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="AIMPackageNotification">
      <soap12:operation soapAction="http://cisco.aim.ns/AIMPackageNotification" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CiscoAIMNotification">
    <wsdl:port name="CiscoAIMNotificationSoap11" binding="tns:CiscoAIMNotificationSoap11">
      <soap:address location="http://localhost:9792"/>
    </wsdl:port>
    <wsdl:port name="CiscoAIMNotificationSoap12" binding="tns:CiscoAIMNotificationSoap12">
      <soap12:address location="http://localhost:9793"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

6 . Export/Import example

The code in italics is used only for the import (AIMPackageNotification) interface. Base clases for some obects are not provided.

```
#include <iostream>
#include "../../lib/CiscoAIMSOAP/CiscoAIMSoap11.nsmap"
#include "soapH.h"
#include "soapStub.h"
#include "pThread.hpp"
#include "pURL.hpp"

using namespace std;

SOAP_FMAC5 int SOAP_FMAC6 __YOUR__AIMPackageNotification(struct soap*, struct
_YOUR__AIMPackageNotification *pIn, struct _YOUR__AIMPackageNotificationResponse *pOut)
{
  int nRC = 0;
  cout << "***SOAP 1.1***PackageNotification incoming*********************" << endl;
  cout << "*** PackageName   : " << pIn->PackageName << endl;
  cout << "*** ADIURL        : " << pIn->ADIURL << endl;
  cout << "*** Ingest result : " << pIn->Result << endl;
  cout << "***SOAP 1.1***PackageNotification complete*********************" << endl;
  pOut->NotificationResult = &nRC;
  return(0);
}

SOAP_FMAC5 int SOAP_FMAC6 __YOUR__AIMPackageNotification_(struct soap*, struct
_YOUR__AIMPackageNotification *pIn, struct _YOUR__AIMPackageNotificationResponse *pOut)
{
  int nRC = 0;
  cout << "***SOAP 1.2***PackageNotification incoming*********************" << endl;
  cout << "*** PackageName   : " << pIn->PackageName << endl;
  cout << "*** ADIURL        : " << pIn->ADIURL << endl;
  cout << "*** Ingest result : " << pIn->Result << endl;
  cout << "***SOAP 1.2***PackageNotification complete*********************" << endl;
  pOut->NotificationResult = &nRC;

  return(0);
}

class NotifyThread : public pThread
{
  public:
    NotifyThread(char *pEndPoint)
    {
      _pEPoint = pEndPoint;
    };
    int _run()
    {
      int    nLSock = -1;
      int    nCount = 1;
      soap_init(&_pSOAP);
      pURL   pURLEndP("SOAP Notify",_pEPoint);


while(!soap_valid_socket(soap_bind(&_pSOAP,pURLEndP.pszHost,pURLEndP.nPort,100))&&(sleep(
2)))
        cout << string("Attempting to bind Notification soap service - count " + pDe-
bug::toStr(nCount++)) << endl;

      cout << string("Notification SOAP bound at "+string(_pEPoint)) << endl;
      while(getStatus() < 6)
      {
        int nASock;
        _pSOAP.send_timeout   = 600;
        _pSOAP.recv_timeout   = 600;
        _pSOAP.max_keep_alive = 1000;

        nASock = soap_accept(&_pSOAP);
        if((nASock < 0)||getStatus() == 6)
        {
          soap_print_fault(&_pSOAP,stdout);
          close(nASock);
          continue;
        }
        else
          cout << string("Accepted soap connection") << endl;

        if(soap_serve(&_pSOAP))
          soap_print_fault(&_pSOAP,stdout);
        soap_destroy(&_pSOAP);
        soap_end(&_pSOAP);
     }
     soap_destroy(&_pSOAP);
     soap_end(&_pSOAP);
     soap_done(&_pSOAP);
  };
  virtual void stop()
  {
    setStatus(6);
    _YOUR__AIMPackageNotification         in;
    _YOUR__AIMPackageNotificationResponse out;

    in.PackageName = "SHUTDOWN";
    in.ADIURL      = "SHUTDOWN";
    in.Result      = "SHUTDOWN";

    struct soap SOAP;
    soap_init(&SOAP);
    if(soap_call___YOUR__AIMPackageNotification(&_pSOAP,NULL,NULL,&in,&out))
      cout << "Error shutting down notification interface!" << endl;
    else
      cout << "Notification interface shutdown" << endl;
    release();
    soap_destroy(&_pSOAP);
    soap_end(&_pSOAP);
    soap_done(&_pSOAP);
  };
  private:
    char        *_pEPoint;
    struct soap  _pSOAP;
};

void __printSOAPerror(soap *pSOAP,int nRC)
{
  cout << "gSOAP error!!" << endl;
  soap_print_fault(pSOAP,stdout);
}

int main(int argc,char *argv[])
{
  char szSOAPEndPoint[1024] = {0x00};
  char szPackageName[128]   = {0x00};
  char szADIURL[512]        = {0x00};
  int  bDoAsync = 0;
  int  bMetaOnly= 0;
  int  nRC      = 0;
  int  nRequest = 0;
  int  nCount   = 1;

  if(argc < 1)
  {
    cout << "Usage:  SOAPClient [(-! test) -Notification Only Mode ] -# request -c [# of
ingests] -s SOAP endpoint -P [PackageName1,PackageName2, ...] -U [ADIURL1, ...] -M [1|0
meta only?] -Y [1|0 do async?]" << endl;
    cout << "                   -# [#] AIM request" << endl;
    cout << "          Cisco  requests" << endl;
    cout << "                       0  Ingest" << endl;
    cout << "                       1  Delete" << endl;
    cout << "                       2  Update" << endl;
    cout << "                       4  GetPackageStatus" << endl;
    cout << "                       8  GetAllPackages" << endl;
    return(-1);
  }
  cout << "**************************************SOAPClient init*****************" <<
endl;
  for(int a=1;a<argc;a++)
  {
    cout << "[" << a << "] argv[" << argv[a] << "][" << argv[a+1] << "]" << endl;
    switch(argv[a++][1])
    {
      case '!' :
        nCount = 0;
        break;
      case '#' :
        nRequest = atoi(argv[a]);
        break;
      case 'c' :
        nCount   = atoi(argv[a]);
        break;
      case 's' :
        sprintf(szSOAPEndPoint,"%s",argv[a]);
        break;
      case 'P' :
        sprintf(szPackageName,"%s",argv[a]);
        break;
      case 'U' :
        sprintf(szADIURL,"%s",argv[a]);
        break;
      case 'M' :
        bMetaOnly = atoi(argv[a]);
        break;
      case 'Y' :
        bDoAsync = atoi(argv[a]);
        break;
    }
  }
  int a=0;
  struct soap pSOAP;
  soap_init(&pSOAP);
  pSOAP.recv_timeout = 1000000000;
  NotifyThread pNT("http://localhost:9793");
  pNT.start();

  _CISCOAIM__IngestPackage         inStruct0;
  _CISCOAIM__IngestPackageResponse outStruct0;
  _CISCOAIM__UpdatePackage         inStruct00;
  _CISCOAIM__UpdatePackageResponse outStruct00;
  _CISCOAIM__DeletePackage         inStruct000;
  _CISCOAIM__DeletePackageResponse outStruct000;
  _CISCOAIM__GetPackageStatus         inStruct1;
  _CISCOAIM__GetPackageStatusResponse outStruct1;
  _CISCOAIM__GetAllPackages         inStruct2;
  _CISCOAIM__GetAllPackagesResponse outStruct2;

  cout << "*************************************init complete*******************" <<
endl;
  pNT.release();
  char *pResult = "";
  cout << "[" << nCount << "] # of iterations" << endl;
  for(int a=0;a<nCount;a++)
  {
    switch(nRequest)
    {
    case 0 :
      inStruct0.ADIURL       = szADIURL;
      inStruct0.MetaDataOnly = &bMetaOnly;
      inStruct0.PackageName  = szPackageName;
      inStruct0.DoAsync      = &bDoAsync;
      cout << "IngestPackage" << endl;
      nRC = soap_call___CISCOAIM__IngestPackage(&pSOAP,NULL,NULL,&inStruct0,&outStruct0);
      pResult = outStruct0.IngestResult;
      break;
    case 1 :
      inStruct00.ADIURL       = szADIURL;
      inStruct00.MetaDataOnly = &bMetaOnly;
      inStruct00.PackageName  = szPackageName;
      cout << "UpdatePackage" << endl;
      nRC =
soap_call___CISCOAIM__UpdatePackage(&pSOAP,NULL,NULL,&inStruct00,&outStruct00);
      pResult = outStruct00.UpdateResult;
      break;
    case 2 :
      inStruct000.PackageName  = szPackageName;
      cout << "DeletePackage" << endl;
      nRC =
soap_call___CISCOAIM__DeletePackage(&pSOAP,NULL,NULL,&inStruct000,&outStruct000);
      pResult = outStruct000.DeleteResult;
      break;
    case 4 :
      inStruct1.PackageName  = szPackageName;
      cout << "GetPackageStatus" << endl;
      nRC =
soap_call___CISCOAIM__GetPackageStatus(&pSOAP,NULL,NULL,&inStruct1,&outStruct1);
      pResult = outStruct1.StatusResult;
      break;
    case 8 :
      cout << "GetAllPackages" << endl;
      nRC =
soap_call___CISCOAIM__GetAllPackages(&pSOAP,NULL,NULL,szPackageName,&outStruct2);
      pResult = "";
      cout << "--Results " << endl;
      for(a;outStruct2.PackageList&&a<outStruct2.PackageList->__sizestring;a++)
        cout << "[" << a << "] " << outStruct2.PackageList->string[a] << endl;
      break;
    default :
      cout << "Invalid request ! " << nRequest << endl;
      break;
    };
    cout << "--SOAPrc [" << nRC << "]" << endl;
    cout << "--Result [" << pResult << "]" << endl;

    if(nRC) __printSOAPerror((soap*)&pSOAP,nRC);
  }
  if(nCount != 0)
  {
    cout << "press any key to exit" << endl;
    getc(stdin);
  }
  else
    sleep(100000000);
  pNT.stop();
  pNT.release();
  cout << "*************************************complete************************" << 
endl;
  return(0);
}
```

7 . Implementation information

Here is a possible make file and options used when generating the stub code.

```
CLIENT_NAME := CiscoSOAPClient
SOURCE      := $(wildcard *.c)
OBJECT      := $(SOURCE:.c=.o)
GSOAP       := ../../soap/gsoap-linux-2.7.9
INC_FLAG    := -I . -I ../../include -I $(GSOAP)/import -I ../../../shared/include
GCC_FLAG    := -w -pipe -O3 -D_POSIX_THREADS -D_POSIX_THREAD_SAFE_FUNCTIONS -D_REENTRANT
-Wno-deprecated -z muldefs
LIB_FLAG    := -L ../../lib
LD_LIBS     := -lpthread -lssl -lstdc++ -lAVSdCore_332 -lAVSUtil_AIM
GCC         := g++ -g

all : client

wsdl:
               @echo "*******generating stubs from wsdl file****************"
               @$(GSOAP)/wsdl2h -c -g -nYOUR CiscoAIM.wsdl CiscoAIMNotification.wsdl
               @echo "*******compiling the generated stub code**************"
               @$(GSOAP)/soapcpp2 -L -x -w -I$(GSOAP)/import CiscoAIM.h

client: $(CLIENT_NAME)
$(CLIENT_NAME): $(OBJECT)
               @echo "**** Building client :" $@
               @echo ""
               $(GCC) $(GCC_FLAG) $(LIB_FLAG) $? -o $@ $(LD_LIBS)
               @echo ""
               @echo "****************   DONE    ********************"

clean :
       @echo ""
       @echo "****************  CLEANING ********************"
       @echo ""
       @echo "REMOVING FILES : *.o *.so"
       @rm -rf *.o  \
       @rm -rf *.nsmap \
       @echo ""
       @echo "****************   DONE    ********************"
       @echo ""

# <=== C++ COMPILING RULES =====>

$(OBJECT) : $(SOURCE)
       @echo "** COMPILING FILE :" $*.c
       $(GCC) $(GCC_FLAG) $(INC_FLAG) $(LIB_FLAG) -c $*.c
       @echo ""
```