



WEBSERVICES INTEGRATION GUIDELINES FOR PCS WEB SERVICES GATEWAY

Release 0.1 Draft – Feb 2009

Confidentiality & Copyright

This Manual contains CrimsonLogic proprietary material. While CrimsonLogic customers are given reasonable opportunity to view the Manual for the purpose of exemplifying CrimsonLogic's commitment to quality, any form of reproduction, transmission or use of this Manual or its contents is not permitted without prior written approval from CrimsonLogic. All rights are reserved.

CrimsonLogic Pte Ltd

31 Science Park Road, The Crimson, Singapore 117611, Main: (65) 6887 7888, Fax: (65) 6778 5277,
www.crimsonlogic.com.sg

Prepared By

Designation	Name	Signature	Date (dd.mm.yyyy)
Senior Systems Analyst	Y.VENKATESAM		09.02.2009

REVIEWED BY

ROLE	NAME	SIGNATURE	DATE (DD.MM.YYYY)
Project Manager	SOMASHEKAR		12.02.2009

Approvals

Designation	Name	Signature	Date (dd.mm.yyyy)
Project Manager	SOMASHEKAR		12.02.2009

Distribution List

Doc Control No.	Date Issued	Recipient	Hardcopy or Softcopy

Release History

Release Number	Date (mm.yyyy)	Brief Summary of changes
0.1	13.02.2009	Draft Version

Table of Contents

1	INTRODUCTION.....	1
1.1	About the Project.....	1
1.2	Purpose and Scope	1
1.3	Abbreviations.....	1
1.4	Web Service Definition.....	1
1.5	References Materials	2
2	PCS WEBSERVICES GATEWAY (PCSWG)	3
2.1	Overview	3
2.2	WebServices Specification.....	3
3	PCS AS SERVICE PROVIDER	5
3.1	Web Services Published by PCS.....	6
3.2	Berthing Details	6
3.3	How to use the Web Services Published by PCS?	12
4	PCS AS SERVICE CONSUMER	13
4.1	How PCS consumes the Web Services Published by Ports?	14

1 INTRODUCTION

1.1 About the Project

Centralized Port Community System (PCS) is an initiative by Indian ports association (IPA) intended to provide a single window system for the port communities in India to securely exchange the documents and information electronically with their stakeholders involved in the maritime transport and logistics chain including the trading partners and government agencies. It also expected to provide visibility and access to the central database to all its stakeholders through internet based interfaces.

1.2 Purpose and Scope

The document describes in detail about the web services and integration guidelines of PCS web services through PCS Web Services Gateway.

The target audience for this document is project team members, Ports and other major Stakeholders who can publish / consume web services through PCS Web Services Gateway.

1.3 Abbreviations

Following are the terms and their definitions as used in this document:

SOAP: Simple Object Access Protocol, as defined by [W3C](#).

WSDL: Web Services Description Language, as defined by [W3C](#).

PCSWG: PCS Web Services Gateway

PCS: Port Community System

SOA: Service Oriented Architecture

1.4 Web Service Definition

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.

The two basic elements of Web services are:

1. Simple Object Access Protocol (SOAP)
2. Web Services Description Language (WSDL)

1.4.1 SOAP

SOAP is a simple XML-based communication protocol and it will allow application to exchange information over HTTP. It is platform, language independent and extensible.

1.4.2 WSDL

WSDL describes what a Web service does, how it communicates, and where it lives. It is xml-based language and W3C standard.

1.5 References Materials

Web Services Description Language (WSDL), [W3C](#)

Simple Object Access Protocol (SOAP), [W3C](#)

Web Services Addressing (WS-Addressing), [W3C](#)

Web Services Security (WS-Security), [W3C](#)

2 PCS WEBSERVICES GATEWAY (PCSWSG)

2.1 Overview

PCS WebServices Gateway (PCSWSG) is used for Web Services and SOA Management. It provides standards based framework for deployment and management of web services.

The PCSWSG enables PCS and Ports to provide, secure, govern and manage web services from a centralized place. The PCSWSG a comprehensive SOA governance framework by allowing users to define and manage the policies globally and enforcing them on the services.

The PCSWSG makes it possible for Port applications to manage their SOA deployment by tracking and controlling the services with an SOA governance framework, thereby providing more control on the service usage and ensuring the quality of service.

PCSWSG is a WS-I compatible framework for Web services deployment and management in PCS. It provides a central and standardized framework for registering and accessing the web services available to any application. The framework provides a robust implementation of web services related specifications (e.g. WS-Addressing, WS-Security etc.) as well as functional components for alerts etc. PCSWSG provides a downloadable client API implementation for every web service, allowing the application developers to delegate the complexity of accessing the web services to the gateway. The Web Services Gateway is also a framework for exposing the existing application services (e.g. EJBs etc.) as Web Service and provides security, reliability to these exposed services. This is achieved by registering the application service components on the gateway.

The PCSWSG has the following main objectives:

1. Allow external Web Services to be registered on the PCSWSG and act as a broker between the actual web service and client application.
2. Provide a client API to access these web services from any stakeholder application.
3. Provide an Implementation of Web Services related specifications.

2.2 WebServices Specification

2.2.1 SOAP

The PCSWSG provides support for SOAP 1.1 specifications as defined by W3C

2.2.2 WSDL

The PCSWSG supports WSDL 1.1 specifications as defined by W3C

2.2.3 INTEROPERABILITY

The gateway conforms to the Interoperability standards. The gateway conforms to the following WS-I specifications:

- a. WS-I Basic Profile 1.1
- b. WS-I Basic Security Profile

2.2.4 ADDRESSING

The gateway supports W3C's WS-Addressing 1.1 specifications.

2.2.5 SECURITY POLICIES

The security constraints are maintained using the security policies. The security policies apply to the external web services registered on the WSG as well as the application services being exposed as web service. The WSG will be able to identify the WS-SecurityPolicy metadata in the WSDL document while registering an external web service and use them while accessing the web service.

The following security mechanisms are supported:

- a. Http digest authentication
- b. Username/password token

While exposing the application service, user can apply various security constraints for accessing the service. This includes a set of standard security mechanism as described in the WS-Security specification. User may choose to apply more than one security constraint on the service, in a specified order as well. E. g. the message can contain a userid/password token to access the service, and the message can be then signed and encrypted.

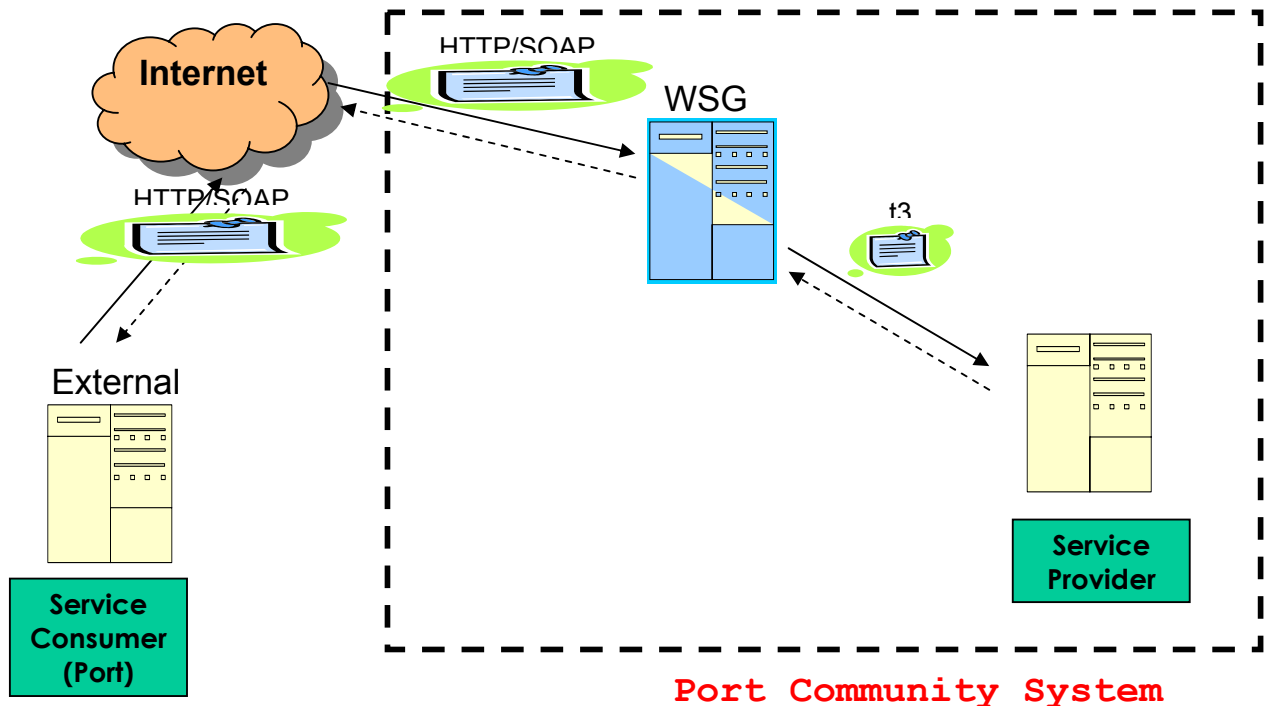
When an application service that is exposed as web service has security policy applied, the generated WSDL for the service will contain WS-SecurityPolicy elements.

3 PCS AS SERVICE PROVIDER

A **service provider** or **publisher** means enabling a Web service user (consumer) to locate the service description and instructing the consumer how they should interact with the Web service.

PCS publishes web services through PCSWSG and provide the WSDL files to all Ports for consuming the service. Ports can download the WSDL files from the PCS Portal and can make a call to a particular service (consume) by sending the SOAP Request using HTTP protocol.

When Ports want to consume a particular web service, they will send the SOAP Request to PCSWSG. PCSWSG communicates to PCS application through t3 protocol and forwards this SOAP request to PCS application. PCS application unpacks the SOAP Request and calls the appropriate method of the specified class. This method when complete returns some value which is packed up and sent back to consumer in an HTTP response.



3.1 Web Services Published by PCS

PCS publishes the following web services through PCSWSG.

- Berthing Details
- Ship Schedules

3.2 Berthing Details

Ports sends the following input parameters as SOAP request to PCSWSG through HTTP protocol to get the Berthing details for a given two dates (From Date and to Date):

1. Port Code
2. From Date
3. To Date

3.2.1 WSDL

This WSDL file provides the URL of the Berthing Details web service and input parameters that consumer should send and also return object which is sent by the producer to the consumer.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:ns4="http://exception.common.ipapcs.trade.crimsonlogic.com/xsd"
xmlns:ns0="http://vo.webservices.ipapcs.trade.crimsonlogic.com/xsd"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:axis2="http://business.webservices.ipapcs.trade.crimsonlogic.com"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:ns1="http://rmi.java/xsd"
xmlns:ns3="http://io.java/xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ns2="http://business.webservices.ipapcs.trade.crimsonlogic.com/xsd"
targetNamespace="http://business.webservices.ipapcs.trade.crimsonlogic.com">
  <wsdl:types>
    <xs:schema
xmlns:ax24="http://vo.webservices.ipapcs.trade.crimsonlogic.com/xsd"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://vo.webservices.ipapcs.trade.crimsonlogic.com/xsd">
      <xs:complexType name="RequestBerthVO">
        <xs:sequence>
          <xs:element minOccurs="0" name="fromDate" nillable="true"
type="xs:string"/>
          <xs:element minOccurs="0" name="portCode" nillable="true"
type="xs:string"/>
          <xs:element minOccurs="0" name="toDate" nillable="true"
type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="ResponseBerthVO">
        <xs:sequence>
          <xs:element minOccurs="0" name="details" nillable="true"
type="xs:anyType"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
    <xs:schema xmlns:ax22="http://rmi.java/xsd" attributeFormDefault="qualified"
elementFormDefault="qualified" targetNamespace="http://rmi.java/xsd">
      <xs:complexType name="RemoteException">
        <xs:complexContent>
          <xs:extension base="ns3:IOException">
            <xs:sequence>
              <xs:element minOccurs="0" name="cause"
nillable="true" type="xs:anyType"/>
              <xs:element minOccurs="0" name="message"
nillable="true" type="xs:string"/>
              <xs:element minOccurs="0" name="detail"
nillable="true" type="xs:anyType"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:schema>
  </wsdl:types>

```

```

        </xs:complexContent>
      </xs:complexType>
    </xs:schema>
  </xs:schema>
  xmlns:xsd="http://business.webservices.ipapcs.trade.crimsonlogic.com/xsd"
  attributeFormDefault="qualified" elementFormDefault="qualified"
  targetNamespace="http://business.webservices.ipapcs.trade.crimsonlogic.com/xsd">
    <xs:complexType name="Exception">
      <xs:sequence>
        <xs:element minOccurs="0" name="Exception" nillable="true"
type="xs:anyType"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="PCSBException">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="PCSBException"
nillable="true" type="ns4:PCSBException"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="RemoteException">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="RemoteException"
nillable="true" type="ns1:RemoteException"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="getWSBerthDetails">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="param0" nillable="true"
type="ns0:RequestBerthVO"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="getWSBerthDetailsResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="return" nillable="true"
type="ns0:ResponseBerthVO"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
  <xs:schema xmlns:ax23="http://io.java/xsd" attributeFormDefault="qualified"
elementFormDefault="qualified" targetNamespace="http://io.java/xsd">
    <xs:complexType name="IOException">
      <xs:complexContent>
        <xs:extension base="ns2:Exception">
          <xs:sequence/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:schema>

```

```

    <xs:schema
xmlns:ax21="http://exception.common.ipapcs.trade.crimsonlogic.com/xsd"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://exception.common.ipapcs.trade.crimsonlogic.com/xsd">
    <xs:complexType name="PCSBASEException">
        <xs:complexContent>
            <xs:extension base="ns2:Exception">
                <xs:sequence>
                    <xs:element minOccurs="0" name="errorCode"
nillable="true" type="xs:string"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
</wsdl:types>
<wsdl:message name="getWSBerthDetailsRequest">
    <wsdl:part name="parameters" element="ns2:getWSBerthDetails"/>
</wsdl:message>
<wsdl:message name="getWSBerthDetailsResponse">
    <wsdl:part name="parameters" element="ns2:getWSBerthDetailsResponse"/>
</wsdl:message>
<wsdl:message name="PCSBASEException">
    <wsdl:part name="parameters" element="ns2:PCSBASEException"/>
</wsdl:message>
<wsdl:message name="RemoteException">
    <wsdl:part name="parameters" element="ns2:RemoteException"/>
</wsdl:message>
<wsdl:portType name="vesselWSDLPortType">
    <wsdl:operation name="getWSBerthDetails">
        <wsdl:input message="axis2:getWSBerthDetailsRequest"
wsaw:Action="urn:getWSBerthDetails"/>
        <wsdl:output message="axis2:getWSBerthDetailsResponse"
wsaw:Action="urn:getWSBerthDetailsResponse"/>
        <wsdl:fault name="PCSBASEException" message="axis2:PCSBASEException"
wsaw:Action="urn:getWSBerthDetailsPCSBASEException"/>
        <wsdl:fault name="RemoteException" message="axis2:RemoteException"
wsaw:Action="urn:getWSBerthDetailsRemoteException"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="vesselWSDLSOAP11Binding" type="axis2:vesselWSDLPortType">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getWSBerthDetails">
        <soap:operation soapAction="urn:getWSBerthDetails" style="document"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="PCSBASEException">
            <soap:fault name="PCSBASEException" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="RemoteException">
            <soap:fault name="RemoteException" use="literal"/>
        </wsdl:fault>
    </wsdl:operation>
</wsdl:binding>
</wsdl:service>
</wsdl:definitions>

```

```

        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="vesselWSDLHttpBinding" type="axis2:vesselWSDLPortType">
      <http:binding verb="POST"/>
      <wsdl:operation name="getWSBerthDetails">
        <http:operation location="vesselWSDL/getWSBerthDetails"/>
        <wsdl:input>
          <mime:content part="getWSBerthDetails" type="text/xml"/>
        </wsdl:input>
        <wsdl:output>
          <mime:content part="getWSBerthDetails" type="text/xml"/>
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="vesselWSDL">
      <wsdl:port name="vesselWSDLSOAP11port_http"
binding="axis2:vesselWSDLSOAP11Binding">
        <soap:address
location="http://tmhub.asianconnect.com/wsg/services/coi/IPAPCSID"/>
      </wsdl:port>
      <wsdl:port name="vesselWSDLHttpport" binding="axis2:vesselWSDLHttpBinding">
        <http:address
location="http://tmhub.asianconnect.com/wsg/services/coi/IPAPCSID"/>
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>

```

3.2.2 REQUEST

Following is the sample SOAP request message for the Berthing Details web service

```

<?xml version="1.0" encoding="utf-8" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header />
  <soapenv:Body>
    <ipapcs:getWSBerthDetails
xmlns:ipapcs="http://business.webservices.ipapcs.trade.crimsonlogic.com/xsd">
      <ipapcs:request xmlns:req="http://vo.webservices.ipapcs.trade.crimsonlogic.com/xsd">
        <req:fromDate>02022008:12:12</req:fromDate>
        <req:portCode>INJNP1</req:portCode>
        <req:toDate>28042008:12:12</req:toDate>
      </ipapcs:request>
    </ipapcs:getWSBerthDetails>
  </soapenv:Body>
</soapenv:Envelope>

```

3.2.3 RESPONSE

Following is the sample SOAP response message for the Berthing Details web service

```

<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>

```

```

    <ipapcs:ResponseBerthVO
xmlns:ipapcs="http://business.webservices.ipapcs.trade.crimsonlogic.com/xsd">
  <ipapcs:return
xmlns:ipapcs2="http://vo.webservices.ipapcs.trade.crimsonlogic.com/xsd">
    <ipapcs:VO>
      <ipapcs2:PortCode>INJNP1</ipapcs2:PortCode>
      <ipapcs2:VCN>0707666</ipapcs2:VCN>
      <ipapcs2:imoNumber>454554</ipapcs2:imoNumber>
      <ipapcs2:ETD>03042008:17:26</ipapcs2:ETD>
      <ipapcs2:ETA>02042008:17:26</ipapcs2:ETA>
      <ipapcs2:berthCode>B09</ipapcs2:berthCode>
      <ipapcs2:vesselName>CHEMSTAR SEVEN</ipapcs2:vesselName>
    </ipapcs:VO>
    <ipapcs:VO>
      <ipapcs2:PortCode>INJNP1</ipapcs2:PortCode>
      <ipapcs2:VCN>1294943586789</ipapcs2:VCN>
      <imoNumber/>
      <ipapcs2:ETD>09042008:15:48</ipapcs2:ETD>
      <ipapcs2:ETA>08042008:15:48</ipapcs2:ETA>
      <berthCode/>
      <vesselName/>
    </ipapcs:VO>
    <ipapcs:VO>
      <ipapcs2:PortCode>INJNP1</ipapcs2:PortCode>
      <ipapcs2:VCN>1294943586789</ipapcs2:VCN>
      <imoNumber/>
      <ipapcs2:ETD>11042008:10:51</ipapcs2:ETD>
      <ipapcs2:ETA>10042008:10:51</ipapcs2:ETA>
      <berthCode/>
      <vesselName/>
    </ipapcs:VO>
    <ipapcs:VO>
      <ipapcs2:PortCode>INJNP1</ipapcs2:PortCode>
      <ipapcs2:VCN>2001</ipapcs2:VCN>
      <imoNumber/>
      <ipapcs2:ETD>03042008:17:03</ipapcs2:ETD>
      <ipapcs2:ETA>02042008:17:03</ipapcs2:ETA>
      <ipapcs2:berthCode>ABCDEF</ipapcs2:berthCode>
      <vesselName/>
    </ipapcs:VO>
    <ipapcs:VO>
      <ipapcs2:PortCode>INJNP1</ipapcs2:PortCode>
      <ipapcs2:VCN>9876543VCNp19</ipapcs2:VCN>
      <imoNumber/>
      <ipapcs2:ETD>05042008:11:12</ipapcs2:ETD>
      <ipapcs2:ETA>04042008:11:12</ipapcs2:ETA>
      <berthCode/>
      <vesselName/>
    </ipapcs:VO>
  </ipapcs:return>
</ipapcs:ResponseBerthVO>
</soapenv:Body>
</soapenv:Envelope>

```

3.3 How to use the Web Services Published by PCS?

When PCS publishes a web service, it will provide its WSDL file to the Ports after registering WSDL file in PCSWSG.

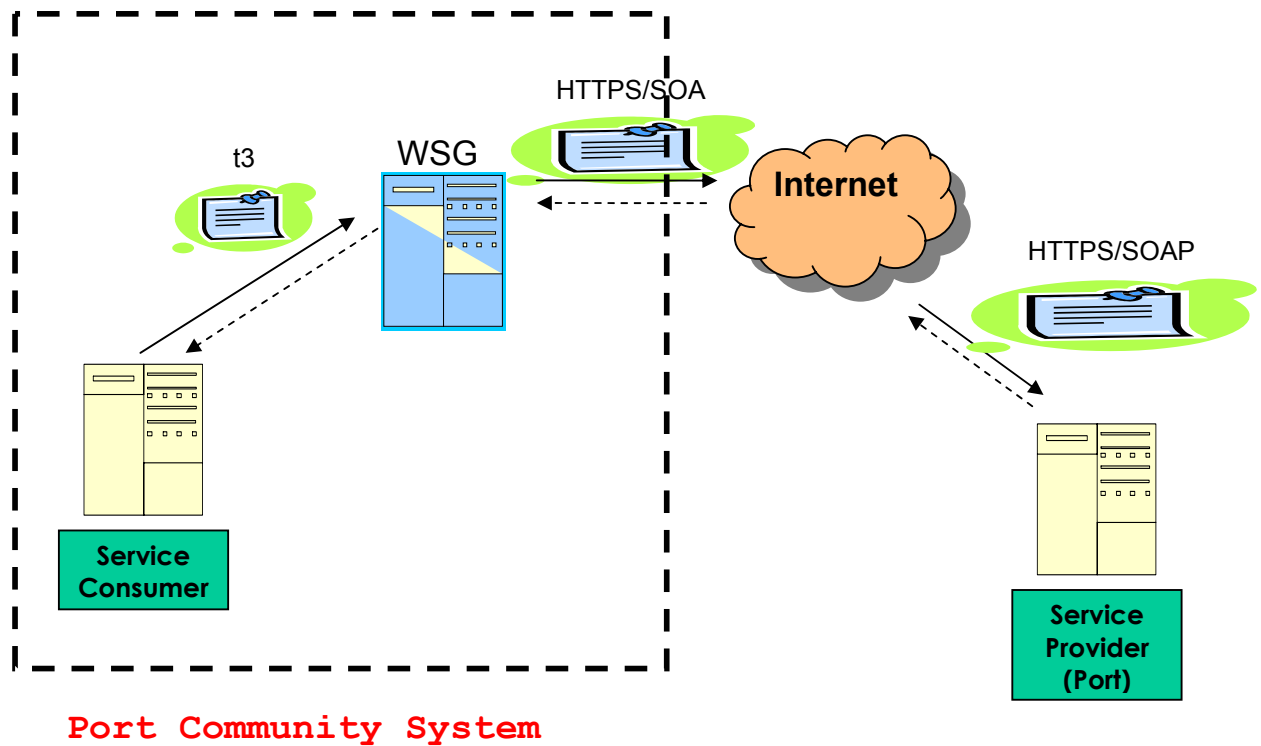
- The following steps are followed while publishing a web service by PCS:
 1. PCS application will create and deploy a web service through PCSWSG
 2. PCSWSG generates WSDL file
 3. Ports can download the WSDL file from the PCS Portal
 4. Ports should generate client API using this WSDL file
 5. Using the Client APIs,
 - 5.1 An HTTP request is made from the Ports to the PCSWSG, specifying the method it wishes to call and passing along the parameters in either a SOAP request, through the QueryString or in the POST headers.
 - 5.2 PCSWSG receives the incoming request and forwards to PCS Application
 - 5.3 PCS application unpackages the input parameters, and calls the appropriate method.
 - 5.4 When completed, this method returns some value which is packages up and sent back to the Port through PCSWSG in an HTTP response.
 - 5.5 The Ports receives this response in the form of SOAP response(xml) and can integrate with their internal application

4 PCS AS SERVICE CONSUMER

A service consumer is a network application to access information and functionality provided by a Web service provider using Internet Protocol.

PCS consumes a web service which is published by Ports through PCSWSG. Ports will generate and provides WSDL file to PCSWSG.

When PCS wants to consume a particular web service, it will send the SOAP Request to Port through PCSWSG. A port web site unpacks the SOAP Request and calls the appropriate method of the specified class. This method when complete returns some value which is packed up and sent back to PCSWSG in an HTTP response. PCSWSG communicates to PCS application using t3 protocol. PCS application unpacks the response object and displays on the PCS Portal.



4.1 How PCS consumes the Web Services Published by Ports?

When Ports publish a web service, it will provide its WSDL file to the PCSWSG.

- o The following steps are followed while publishing a web service by Ports:
 1. Ports application will create and deploy a web service through their web sites
 2. Ports application generates WSDL and provides to the PCS
 3. PCSWSG register this WSDL file and generates a client API and provides to PCS application.
 4. Using the Client APIs,
 - 4.1 A SOAP request is made by the PCS application using t3 protocol and send the request to PCSWSG and WSG makes an HTTP request to the Ports web site
 - 4.2 Port(s) receives the incoming request and unpacks the input parameters and calls the appropriate method.
 - 4.3 When completed, this method returns some value which is packages up and sent back to the PCSWSG in an HTTP response.
 - 4.4 The PCS receives this response in the form of SOAP response(xml) and parse the xml file and displays on the PCS Portal.