منتدى الهندسة المعلوماتية
سوريا
حلب

Simple Object Access Protocol
&
Web Services Description Language

فادي عمروش
alfady@scs-net.org

محمد ناشد
alnashed@scs-net.org

جميع الحقوق محفوظة©
الثلاثاء، 31 أيار، 2005
www.aleposoft.com

الفهرس

مقدمة (Introduction):

تعتبر خدمات ويب من أكثر الخدمات انتشارات في وقتنا الحالي و حققت خدمات ويب قفزة كبيرة في طريقة إنجاز الانترنت للأعمال و أصبح استعراض صفحات الويب لإدخال أوامر أمرا جيدا للتطبيقات التجارية و خاصة تطبيقات المستهلكين B2C Business to consumer و كمثال عليها موقع يقدم خدمة بيع الكتب للمستهلكين حيث يدخل المستهلك للموقع و يقدم طلبه بالبحث عن كتاب معين فإذا أعجبه يمكن ان يقدم طلب شراء و يشتريه دون عناء الذهاب للمكتبة .و باختصار تحقق لنا خدمات ويب حل المسألة التالية : يستطيع برنامج المستخدم إرسال طلب إلى خدمة ويب التي تقوم بتنفيذ الطلب و إعادة النتيجة له .

1.بروتوكول الوصول البسيط للعناصر (Simple Object Access Protocol):

برتوكول بسيط يعتمد على لغة التوصيف XML وذلك من اجل تبادل المعلومات فوق برتوكول HTTP . وهو برتوكول اتصالات ويستخدم للاتصال بين التطبيقات ومن اجل إرسال الرسائل بينهم, وهو مستقل عن نظام التشغيل (Platform Independent) , ويسمح لك بالالتفاف على الجدار الناري , وهو قياسي وتدعمه المنظمة W3C

1.1.لماذا يستخدم SOAP (Why SOAP)؟

عادة يتم الاتصال بين التطبيقات باستخدام المناهج البعيدة (RPC) بين الصفوف مثل DCOM , CORBA , ولكن برتوكول HTTP ليس مصمم من أجل ذلك , وبالتالي RPC مناسبة من أجل مشكلة الأمن فالجدر النارية ومخدمات البروكسي سوف تعيق مرور مثل هذا النوع من التدفق.

وبالتالي الطريقة الأفضل للاتصال بين التطبيقات هي فوق البرتوكول HTTP, حيث أنه مدعوم من كل مستعرضات الانترنت والمخدمات , وقد تم إنشاء البرتوكول SOAP لإنجاز هذه المهمة (الاتصال فوق البرتوكول HTTP). وهو يؤمن طريقة للاتصال بين تطبيقات تعمل على أنظمة تشغيل مختلفة , وبتقانيات ولغات برمجة مختلفة

2.1.بناء رسالة برتوكول SOAP (SOAP Building Blocks):

رسالة البرتوكول SOAP هي عبارة عن ملف XML يتضمن العناصر التالية :

- عنصر التغليف Envelope المطلوب والذي يعرف ملف XML كرسالة SOAP

- عنصر الترويسة (اختياري) والذي يحتوي معلومات الترويسة
- عنصر المكون الرئيسي Body والذي يحتوي معلومات الاستدعاء والإجابة
- عنصر الخطأ (اختياري) والذي يؤمن معلومات حول الأخطاء التي حدثت أثناء معالجة الرسالة

3.1.قواعد البرتوكول (Syntax Rules):

- رسالة SOAP يجب أن تشفر باستخدام XML
- رسالة SOAP يجب أن تكون مغلفة Enveloped
- رسالة SOAP يجب أن تكون مرمزة Encoded
- رسالة SOAP يجب أن لا يحتوي تعليمات معالجة XML

Skeleton SOAP Message

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  ...
  ...
  <soap:Fault>
```

```
  ...

  ...
 </soap:Fault>
</soap:Body>
</soap:Envelope>
```

إن عنصر تغليف رسالة SOAP هو العنصر الرئيسي والأساس لرسالة SOAP وهو الذي يجعل ملف الـــ XML رسالة SOAP . وبكلام آخر هو الذي يميز ملف XML بحيث نعرفه كرسالة SOAP ونقوم بذلك عن طريق xmlns:soap و التي يمللك القيمة

xmlns:soap="http://www.w3.org/2001/12/soap-envelope"

وبهذه الطريقة نكون قد حددنا التغليف بأنه تغليف SOAP. حيث أنه يجب تحديد NameSpace نفسه فإذا تم تحديد NameSpace مختلف فالتطبيق سيولد خطأ ويلغى الرسالة .

4.1.عناصر ترويسة بروتوكول SOAP الاختيارية (Optional SOAP Header Eelement):

عنصر الترويسة هو عنصر اختياري ويحتوي معلومات محددة عن رسالة SOAP تهم التطبيق (مثل الأذن والشهادة الأمنية ومعلومات عن عمليات الدفع ((PAYMENT)

إن المثال في الأعلى تحتوي فيه الترويسة (Header) على العنصر "Trans" والواصفة "mustUnderstand" ذات القيمة "1" , والقيمة 234

يعرف SOAP ثلاث واصفات في nameSpace الافتراضي

("http://www.w3.org/2001/12/soap-envelope") وهذه الواصفات هي:

- actor
- mustUnderstand
- encodingStyle

واللواتي تحدد كيف ستتم عملية الاستقبال

1.4.1.الواصفة Actor (Actor Element):

إن رسالة soap تنتقل من المرسل إلى المستقبل عبرة طريق طويل قد تمر به بعدة محطات ولذلك فإن مهمة هذه الواصفة هي تحديد المحطة النهائية .

Syntax

```
soap:actor="URI"
```

Example

```
<soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:actor="http://www.w3schools.com/appml/">
234
</m:Trans>
</soap:Header>
```

2.4.1.الواصفة mustUnderstand (mustUnderstand Element):

وتستخدم هذه الواصفة لتحديد فيما إذا الترويسة يجب على المستقبل أن يعالجها أم لا بمعنى آخر هل (إجباري أم اختياري)

- إذا كانت 0 فهو اختياري
- إذا كانت 1 فهو إجباري

Syntax

```
soap:mustUnderstand="0|1"
```

Example

```
soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:mustUnderstand="1">
234
</m:Trans>
</soap:Header>
```

3.4.1.الواصفة encodingStyle (encodingStyle Element):

أما الواصفة encodingStyle تستخدم لتحديد نمط المعطيات المستخدم في الملف. وهذه الواصفة قد تظهر مع أي عنصر SOAP وهي تحديد نوع معطيات ذلك العنصر ومعطيات كل أبنائه أيضاً.

Syntax

```
soap:encodingStyle="URI"
```

Example

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
```

```
...
Message information goes here
...
</soap:Envelope>
```

عنصر الجسم Body soap وهو الذي يحتوي رسالة soap الفعلية.

5.1.عناصر جسم بروتوكول SOAP (The SOAP Body Element):

```
<soap:Body>
    <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
        <m:Item>Apples</m:Item>
    </m:GetPrice>
</soap:Body>
```

المثال السابق يطلب سعر التفاح, عن طريق Getprice وهي ليس من عناصر soap الرئيسة , وبالتالي حددنا أولا صفحة الأسعار Prices ومن ثم عن طريق Item نحدد ما نريد وطلبنا سعر التفاح . وستكون الاستجابة كما يلي

```
<soap:Body>
    <m:GetPriceResponse  xmlns:m="http://www.w3schools.com/prices">
        <m:Price>1.90</m:Price>
    </m:GetPriceResponse>
</soap:Body>
```

6.1.بروتوكول HTTP (The HTTP Protocol):

يعتبر برتوكول HTTP من البرتوكولات التي تعمل فوق برتوكول TCP/IP وذلك لأن هذا البرتوكول يعمل في طبقة التطبيقات , بينما برتوكول TCP/IP فهو في الطبقة الثالثة والرابعة من طبقات تصنيف ISO. فالبرتوكول HTTP يستخدم البرتوكول TCP للإتصال مع مخدم TTP , حيث يرسل الزبون رسالة طلب HTTP إلى المخدم على الشكل التالي:

```
POST /item HTTP/1.1

Host: 189.123.345.239

Content-Type: text/plain

Content-Length: 200
```

وعندئذٍ المخدم يعالج الطلب ويرسل الاستجابة HTTP إلى الزبون.

```
200 OK

Content-Type: text/plain

Content-Length: 200
```

في المثال السابق المخدم يعيد a status code وهو 200 , وهذا دليل نجاح عملية الإرسال والاستقبال حيث أن المخدم إذا لم يفهم الطلب فسوف يعيد غير هذه القيمة وتكون القيمة المعادة قريبة من الشكل التالي

```
400 Bad Request

Content-Length: 0
```

7.1.ربط برتوكول SOAP مع بروتوكول HTTP (SOAP HTTP Binding):

إن المنهاج العام Soap عبارة عن عملية طلب وعملية استجابة HTTP لهذا الطلب والتي تخضع لقوانين Soap التي تتوافق مع XML وبالتالي يصبح لدينا :

HTTP + XML = SOAP

علما أن طلب Soap هو من النوعين:

HTTP POST request
HTTP GET request

ويوجد وصفتين الترويسة HTTP هما Content-Type and Content-Length.

```
Content-Length: bytes
```

Example

```
POST /item HTTP/1.1

Content-Type: application/soap+xml; charset=utf-8

Content-Length: 250
```

حيث عن طريق Content-Type يتم تحديد النمط وعن طريق Content-Length يتم تحديد عدد البايتات لجسم soap

مثال عام:

The SOAP request:

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

A SOAP response:

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

2.لغة وصف خدمات الوب WSDL (Web Services Description Language):

إن لغة وصف خدمات الوب WSDL (Web Services Description Language) عبارة عن لغة مبنية على ترميز XML لوصف خدمات الوب و الانترنت و تسهيل الوصول و التعامل معها .

سيناقش هذا القسم الأفكار التالية :

- ما هي لغة WSDL : و يتحدث عن ماهية هذه اللغة .
- مستندات WSDL : و يتحدث عن الأجزاء الرئيسة لمستند مكتوب بلغة WSDL
- منافذ WSDL: و يتحدث واجهة المنفذ للغة WSDL أي WSDL port interface
- روابط WSDL : و يتحدث عن واجهات الربط للغة WSDL أي WSDL binding interface
- قواعد لغة WSDL: و يعرض القواعد الكاملة للغة WSDL كما جاءت في وثائق W3C

1.2.ما هي لغة WSDL (What is WSDL):

إن لغة وصف خدمات WSDL (Web Services Description Language) عبارة عن لغة مبنية على ترميز XML لوصف خدمات الوب و الانترنت و تسهيل الوصول و التعامل معها .قبل دراسة لغة WSDL عليك أن تكون مطلعا على المبادئ الأساسية للغة XML و خاصة موضوعي Namespaces و Schema.

و لها عدة مميزات أهمها:

- تستخدم لتوصيف خدمات الويب .
- مكتوبة بـــ XML.
- عبارة عن مستند XML
- تقنية موثقة من قبل W3C و لم تصبح معيارية بعد .

إذا إن لغة WSDL هي لغة غير معيارية لتوصيف خدمات الوب و هي عبارة عن مستند مكتوب بلغة XML بحيث يقوم هذا المستند بوصف خدمات الوب كأن يقوم مثلا بتحديد موقع خدمة الوب و طرق و عمليات هذه الخدمة .

و لنتعرف على التطور التاريخي للغة WSDL ضمن W3C حيث قدمت النسخة الأولى WSDL 1.1 من هذه اللغة كمذكرة W3C مقدمة من قبل شركات Ariba, IBM and

Microsoft لتوصيف الخدمات معتمد على بروتوكولات XML التي أوجدت آذار 2001. حيث أن مذكرات W3C هي عبارة عن مذكرات تقوم مؤسسة W3C بجعلها متاحة للجميع للنقاش حولها و تطويرها من قبل عدة شركات و مؤسسات بحث معا و نشر المذكرات بواسطة W3C لا يعني ضرورة أن تكون قد كتبت من قبل المؤسسة ذاتها أو بواسطة فريق المؤسسة أو أحد أعضائها كما تشير مطبوعاتها.صدرت أولى النسخ غير المعيارية Draft للإصدار WSDL 1.2 من قبل W3C في تموز 2002 .

2.2.مستند WSDL (WSDL Document):

بكل بساطة يمكنك القول أن مستند WSDL هو مستند بسيط مكتوب كمستند XML و يحتوي على مجموعة من التعاريف التي تعرف خدمة ويب .

1.2.2.بنية مستند WSDL (WSDL Document Structure):

يعرف مستند WSDL خدمة ويب ما باستخدام العناصر الرئيسية التالية:

العنصر (Element)	الوصف
<portType>	يقوم بتعريف العمليات التي تقوم خدمة ويب ما بتنفيذها .
<message>	يقوم بتعريف الرسائل messages المستخدمة من قبل خدمة ويب ما .
<types>	يقوم بتعريف أنماط المعطيات data types المستخدمة من قبل خدمة ويب ما .
<binding>	يقوم بتعريف بروتوكولات الاتصال المستخدمة من قبل خدمة ويب ما .

و تبدو البنية الرئيسية لمستند WSDL على النحو التالي:

```
<definitions>
<types>
  definition of types.....
</types>
<message>
  definition of a message....
</message>
<portType>
  definition of a port.....
</portType>
<binding>
  definition of a binding....
</binding>
</definitions>
```

يمكن لمستند WSDL أن يحتوي أيضا عناصر أخرى مثل عناصر التوسعة extension elements مثلا أو عنصر خدمة يجمع عدة تعريفات لخدمات ويب متعددة معا ضمن مستند WSDL مفرد ,يمكنك الإطلاع على القواعد الكاملة للغة WSDL من خلال فقرة قواعد WSDL .

2.2.2.منافذ WSDL (WSDL Ports):

يعتبر المحدد <portType> من أحد أهم المعرفات ضمن مستند WSDL , يقوم هذا العنصر بتعريف خدمة ويب , العمليات التي سمكن تنفيذها , الرسائل التي يمكن استخدامها

.يمكن أن تعتبر العنصر <portType> كمثل مكتبة أو صف أو وحدة ضمن لغات البرمجة التقليدية .

3.2.2.رسائل WSDL (WSDL Messages):

يقوم العنصر <message> بتعريف الرسائل messages المستخدمة من قبل خدمة ويب ما حيث يقوم بتعريف عناصر المعطيات لعملية ما .يمكن لرسالة ما أن تتألف من جزء واحد أو أكثر , يمكنك تشبيه أجزاء الرسالة كبرامترات تابع ما ضمن لغات البرمجة التقليدية

4.2.2.أنماط WSDL (WSDL Types):

يقوم العنصر <types> بتعريف أنماط المعطيات data types المستخدمة من قبل خدمة ويب ما و من أجل أمور التوافقية مع كل اللغات الأخرى تستخدم لغة WSDL نفس قواعد XML Schema لتعريف أنماط المعطيات.

5.2.2.روابط WSDL (WSDL Binding):

يقوم العنصر <binding> بتعريف بروتوكولات الاتصال المستخدمة من قبل خدمة ويب ما كما يقوم بتعريف صيغة الرسالة و يقوم بذلك من أجل كل منفذ .

مثال :

فيما يلي مثال يوضح جزء من مستند WSDL:

```
  <part name="value" type="xs:string"/>

</message>

<portType name="glossaryTerms">
 <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
 </operation>
</portType>
```

في المثال السابق يقوم العنصر portType بتعريف "glossaryTerms" كاسم للمنفذ كما يعرف عملية ضمنه باسم : "getTerm" , كما يتم تعريف العملية "getTerm" التي لها رسالة دخل بالاسم "getTermRequest" و رسالة خرج بالاسم "getTermResponse".تعرف عناصر الرسالة message أجزاء كل رسالة و انماط المعطيات الموافقة لها .بالمقارنة مع لغات البرمجة التقليدية يعتبر المنفذ glossaryTerms كمكتبة و "getTerm" تابع ضمن المكتبة له متحول دخل هو "getTermRequest"و متحول خرج هو "getTermRequest".

3.2.منافذ WSDL (WSDL Ports):

يقوم منفذ WSDL بوصف الواجهات (العمليات القانونية) التي سيتم استدعائها من قبل خدمة ويب , كما ذكرنا سابقا فإن العنصر <portType> هو أكثر عناصر لغة WSDL أهمية حيث يقوم هذا العنصر بتعريف خدمة ويب و العمليات التي يمكن إنجازها و الرسائل المتبادلة .يعرف المنفذ نقطة اتصال لخدمة ويب و يمكنك تشبيهه بمكتبة أو صف أو وحدة في لغة برمجة تقليدية و كما ذكرنا سابقا يمكن تشبيه كل عملية ضمنه بالتابع ضمن لغة برمجة تقليدية .
يعتبر النوع طلب_ استجابة request-response النوع الأكثر شيوعا و استخداما و لكن مع ذلك تعرف لغة WSDL الانماط الأربعة التالية :

النوع	التعريف
One-way	اتجاه واحد : يمكن للعملية استقبال رسالة و لكن لن تعيد استجابة .
Request-response	طلب_ استجابة: يمكن للعملية استقبال رسالة و سوف تعيد استجابة .
Solicit-response	استجداء – استجابة : يمكن للعملية إرسال طلب و من ثم تقوم بالانتظار لاستجابة .
Notification	إشعار: يمكن للعملية إرسال رسالة و سوف لن تقوم بالانتظار لاستجابة

1) عملية من النوع اتجاه واحد One-Way :

فيما يلي مثال على عملية ذات اتجاه واحد :

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

في المثال السابق يقوم المنفذ "glossaryTerms" بتعريف عملية باتجاه واحد تدعى "setTerm" تسمح هذه العملية بإدخال مصطلحات جديدة للقاموس باستخدام رسالة "newTermValues". الذي تأخذ برامترات دخل هي "term" و "value" و لا يتم تعريف أي خرج لهذه العملية.

2) عملية طلب– استجابة Request-Response :

فيما يلي مثال عن هذا النوع من العمليات :

جيث في هذا المثال يقوم المنفذ "glossaryTerms" بتعريف عملية من النوع طلب – استجابة و هي "getTerm". , تقوم هذه العملية بإيجاد القيمة المقابلة لمصطلح ما تتطلب هذه العملية رسالة دخل تدعى "getTermRequest" و التي لها المتحول "term" و سوف تعيد لنا رسالة خرج "getTermResponse" مع متحول يدعى "value" .

```
<message name="getTermRequest">

  <part name="term" type="xs:string"/>

</message>

<message name="getTermResponse">

  <part name="value" type="xs:string"/>

</message>

<portType name="glossaryTerms">
 <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
 </operation>
</portType>
```

4.2.روابط WSDL (WSDL Binding):

كما ذكرنا يقوم العنصر <Binding> بتعريف بروتوكولات الاتصال المستخدمة من قبل خدمة ويب ما كما يقوم بتعريف صيغة الرسالة و يقوم بذلك من أجل كل منفذ .

لنأخذ المثال التالي الذي يحتوي عملية طلب – استجابة .

```
<message name="getTermRequest">

  <part name="term" type="xs:string"/>

</message>

<message name="getTermResponse">

  <part name="value" type="xs:string"/>

</message>

<portType name="glossaryTerms">
 <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
```

```
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">

<soap:binding style="document"

transport="http://schemas.xmlsoap.org/soap/http" />

  <operation>

    <soap:operation

     soapAction="http://example.com/getTerm"/>

    <input>

      <soap:body use="literal"/>

    </input>

    <output>

      <soap:body use="literal"/>

    </output>

  </operation>

</binding>
```

يمتلك العنصر binding واصفتين هما واصفة الاسم name , واصفة النوع type.تعرف واصفة الاسم name اسم عملية الربط و طبعا يمكنك استخدام الاسم الذي تريده أما واصفة النوع type تشير المنفذ المطلوب لعملية الربد طبعا في حالتنا المنفذ "glossaryTerms".يمتلك العنصر soap:binding واصفتين أيضا : واصفة التصميم style , واصفة النقل transport .يمكن أن تكون واصفة التصميم style من الشكل "rpc" أو من الشكل "document" و في حالتنا استخدمنا للتصميم الحالة الأخيرة document, أما واصفة النقل HTTP فتقوم بتعريف البروتوكول المستخدم مع SOAP و في حالتنا البروتوكول المستخدم هو HTTP.يقوم العنصر

operation بتعريف كل عملية يعرضها المنفذ .من أجل كل عملية يتم تعريف حدث SOAP الموافق و يجب أن نحدد أيضا كيفية ترميز الدخل و الخرج و في حالتنا استخدمنا "literal" .

5.2.قواعد WSDL (WSDL Syntax):

فيما يلي القواعد الكاملة للإصدار WSDL Syntax كما تم توصيفه في الورقة الغير رسمية Draft الصادرة عن W3C:

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri">
  <import namespace="uri" location="uri"/> *
  <wsdl:documentation .... /> ?
  <wsdl:types> ?
    <wsdl:documentation .... /> ?
    <xsd:schema .... /> *
  </wsdl:types>
  <wsdl:message name="ncname"> *
    <wsdl:documentation .... /> ?
    <part name="ncname" element="qname"? type="qname"?/> *
  </wsdl:message>
  <wsdl:portType name="ncname"> *
    <wsdl:documentation .... /> ?
    <wsdl:operation name="ncname"> *
      <wsdl:documentation .... /> ?
      <wsdl:input message="qname"> ?
```

```
            <wsdl:documentation .... /> ?
        </wsdl:input>
        <wsdl:output message="qname"> ?
            <wsdl:documentation .... /> ?
        </wsdl:output>
        <wsdl:fault name="ncname" message="qname"> *
            <wsdl:documentation .... /> ?
        </wsdl:fault>
    </wsdl:operation>
</wsdl:portType>
<wsdl:serviceType name="ncname"> *
    <wsdl:portType name="qname"/> +
</wsdl:serviceType>
<wsdl:binding name="ncname" type="qname"> *
    <wsdl:documentation .... /> ?
    <-- binding details --> *
    <wsdl:operation name="ncname"> *
        <wsdl:documentation .... /> ?
        <-- binding details --> *
        <wsdl:input> ?
            <wsdl:documentation .... /> ?
```

```
            <-- binding details -->
        </wsdl:input>
        <wsdl:output> ?
            <wsdl:documentation .... /> ?
            <-- binding details --> *
        </wsdl:output>
        <wsdl:fault name="ncname"> *
            <wsdl:documentation .... /> ?
            <-- binding details --> *
        </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ncname" serviceType="qname"> *
    <wsdl:documentation .... /> ?
    <wsdl:port name="ncname" binding="qname"> *
        <wsdl:documentation .... /> ?
        <-- address details -->
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```