9. Веб-сервисы

9.1 Веб-сервисы первого поколения

XML Веб-сервисы (Веб-службы) – это программные компоненты, с помощью которых можно создавать независимые масштабируемые слабосвязанные приложения.

В основе технологии веб-сервисов лежит процесс обмена сообщениями в формате XML-документов. Передача сообщений происходит с использованием протоколов *HTTP, XML, XSD, SOAP, WSDL, UDDI.*

Спецификация определяет три основных стандарта, используемых для поддержки представления, поиска и обмена информацией между веб-сервисами – это WSDL, UDDI и SOAP, образующие так называемый «треугольник СОА». Процесс взаимодействия между клиентом и поставщиком веб-сервиса представлен на рисунке 28.

Рис. 28. Процесс взаимодействия между клиентом и поставщиком веб-сервиса

Протокол *SOAP* ***(уже не*** *«Simple Object Access Protocol», см. ниже)* предназначен для организации взаимодействия удаленных систем при помощи асинхронного обмена XML-отформатированными документами, состоящими из трех частей: конверта (обертки), заголовка и тела. SOAP формирует базовый слой стека протоколов веб-сервисов, обеспечивая инфраструктуру обмена сообщениями между ними. Хотя изначально название протокола SOAP и расшифровывалось как «Простой Протокол Доступа к Объектам» («Simple Object Access Protocol»), в версии 1.2 стандарта от этого определения отошли, и теперь этот акроним ничего не означает.

Язык *WSDL (Web Services Description Language)* описывает сервисы в виде неких абстрактных ресурсов, способных принимать на вход документы определенных типов и инициировать отправление документов других типов. WSDL

используется для описания веб-сервисов и для определения их расположения. WSDL написан на XML и является XML-документом.

WSDL определяет сервис с двух точек зрения: *абстрактной* и *конкретной*. *На абстрактном уровне* сервис задается в терминах посылаемых и принимаемых им сообщений, которые описываются средствами XML Schema в виде, независимом от конкретного транспортного протокола. *На конкретном уровне* определяются привязки к транспортным форматам и точкам физического размещения.

Группа спецификаций WSDL 2.0 состоит из трех основных документов – WSDL Part 1: Core language («Основной язык»), WSDL Part 2: Message exchange patterns («Шаблоны обмена сообщениями»), WSDL Part 2: Bindings («Привязки»).

Протокол *UDDI (Universal Description Discovery & Integration)* представляет собой стандарт на внутреннее устройство и внешние интерфейсы базы данных (репозитория), хранящей описания сервисов. Все описания в БД хранятся в виде XML-записей. Последняя версия UDDI (3.01) обеспечивает репликацию репозиториев со сложными моделями их подчиненности друг другу, построение репозитория из нескольких узлов (и репликацию данных между ними), глобальную уникальность записей и ключей, API публикации описаний и подписки на изменения, средства обеспечения целостности данных, интернационализации записей, шифрования содержимого. В то время как версия UDDI 2.0 предназначалась для поддержки каталогов электронного бизнеса, версия 3.0 ориентирована и на внутрикорпоративное использование для построения корпоративных систем в рамках идеологии сервис-ориентированной архитектуры. Поэтому она допускает создание реестров нескольких типов (общедоступный, частный и с разделяемым доступом).

Архитектуру веб-сервисов составляет масса протоколов и спецификаций. Их можно разбить на четыре части (процесс, описание, сообщения, связь), образующие стек протоколов, в котором каждый верхний уровень опирается на нижний уровень.

Рис. 29. Стек протоколов веб-сервисов

9.2 Стандарт WSDL

Рассмотрим пример простейшего веб-сервиса. Данный сервис будет обеспечивать возведение в квадрат переданного числа. На языке Java этот сервис можно было бы описать в виде следующего исходного кода.

```
public class MyMath
{
     public int squared(int x)
     {
          return x * x;
     }
}
```

Рис. 30. Пример исходного кода для тестового веб-сервиса

Предположим, что данный веб-сервис мы будем размещать в интернете по адресу http://supercomputer.susu.ru/MyMath.

Для того чтобы любой клиент мог получить информацию о том, какие методы предоставляет веб-сервис и как к ним необходимо обращаться, используется WSDL-документ , описывающий полную спецификацию методов взаимодействия с указанным сервисом. Соответственно, нам необходимо сформировать (вручную или автоматически) WSDL-документ и поместить его по адресу http://supercomputer.susu.ru/MyMath.wsdl.

Как уже было сказано в начале главы, стандарт WSDL обеспечивает описание веб-сервиса в виде сообщений, которые может отправить или же принять веб-сервис. Также, он отвечает за методы связывания данных сообщений с ба-

зовой средой передачи данных (чаще всего используется протокол HTTP). В связи с этим выделяют следующие *элементы WSDL-документа:*

- *Блок types* – типы данных, используемые веб-сервисом;
- *Блок message* – сообщения, используемые веб-сервисом;
- *Блок portType* – методы, предоставляемые веб-сервисом;
- *Блок binding* – протоколы связи, используемые веб-сервисом.

9.2.1 Порты WSDL <portType>

Элемент `<portType>` является наиболее важным элементом WSDL. Он определяет сам веб-сервис, предоставляемые им операции и используемые сообщения. Этот элемент можно сравнить с библиотекой функций, в которой указаны входные параметры и результаты работы функции. Рассмотрим, каким образом будет описываться наш сервис “`MyMath`”:

```
<wsdl:portType name="MyMath">
     <wsdl:operation name="squared" parameterOrder="in0">
          <wsdl:input message="impl:squaredRequest“
        name="squaredRequest"/>
          <wsdl:output message="impl:squaredResponse“
       name="squaredResponse"/>
     </wsdl:operation>
</wsdl:portType>
```

Рис. 31. Пример описания блока portType

Как видно из описания, в блоке `portType` описан интерфейс нашего сервиса `MyMath`, состоящий из одной операции `squared`. Также из описания видно, что данная операция должна выполняться с одним параметром `in0` (его описание будет дано в следующем разделе WSDL-документа, блоке `<message>`). Наша операция состоит из двух сообщений: входного (`wsdl:input message="impl:squaredRequest"`) и выходного (`wsdl:output message= "impl:squaredResponse"`). Входное сообщение `"squaredRequest"` – это то сообщение, которое должен передать пользователь веб-сервиса в наш сервис для того, чтобы запустить операцию возведения в квадрат. Выходное сообщение `"squaredResponse"` – это сообщение, которое будет возвращено пользователю после того, как наша операция возведения в квадрат успешно завершится. Интерфейс этих сообщений будет описан в блоке `<message>` WSDL-документа.

9.2.2 Сообщения WSDL <message>

Элемент `<message>` определяет элементы данных операции. Каждое сообщение может содержать одну или несколько частей. Эти части можно сравнить с параметрами вызываемых функций в традиционных языках программирования.

```
    <wsdl:message name="squaredRequest">
          <wsdl:part name="in0" type="xsd:int"/>
     </wsdl:message>
     <wsdl:message name="squaredResponse">
          <wsdl:part name="squaredReturn" type="xsd:int"/>
     </wsdl:message>
```

Рис. 32. Пример описания блока message

Как можно заметить, в блоке `<message>` приводится описание всех частей сообщений `"squaredRequest"` и `"squaredResponse"`, интерфейс которых мы описали ранее в блоке `<portType>`. Каждая часть сообщения – это параметр вызова метода нашего сервиса. В соответствии с этим легко расшифровать, что для проведения нашей операции возведения в квадрат, пользователь должен передать нам 1 входной параметр `"in0"` в виде целого числа, что определяется атрибутом `type="xsd:int"`. Результатом же операции также будет целое число, что описано в блоке `<wsdl:part name="squaredReturn" type="xsd:int"/>`.

9.2.3 Связи WSDL <binding>

Элемент `<binding>` определяет формат сообщения и детали протокола для каждого порта. Он отвечает за то, каким образом элементы абстрактного интерфейса в блоке `<portType>` преобразуются в массивы информации в формате протоколов взаимодействия, например SOAP.

```
    <wsdl:binding name="MyMathSoapBinding" type="impl:MyMath">
      <wsdlsoap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="squared">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="squaredRequest">
               <wsdlsoap:body  encod-
ingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="http://DefaultNamespace" use="encoded"/>
               </wsdl:input>
          <wsdl:output name="squaredResponse">
               <wsdlsoap:body
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="http://DefaultNamespace" use="encoded"/>
          </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
```

Рис. 33. Пример описания блока binding

Основной частью блока `<binding>` является элемент `<soap:binding>`, определяющий конкретный протокол передачи данных. Атрибут `style` определяет тип запроса и может иметь два значения: `"rpc"` и `"document"`. Также, в блоке `<soap:binding>` является `transport`, определяющий протокол, на основе которого будет производиться взаимодействие (обычно HTTP). Внутри

<soap:operation> находится элемент, который описывает значение поля soapAction HTTP-запроса (если вы знакомы с SOAP). Элементы input и output определяют как будут декодироваться входные и выходные сообщения этой операции.

9.2.4 Блоки <port> и <service>

В данных блоках происходит определение, где находится сервис: port – описывает расположение и способ доступа к конечной точке, service – именованная коллекция портов.

```
<wsdl:service name="MyMathService">
      <wsdl:port binding="impl:MyMathSoapBinding“ name="MyMath">
            <wsdlsoap:address
            location="http://supercomputer.susu.ru/MyMath"/>
      </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Рис. 34. Пример блока service

Как можно видеть из примера, в блоке <port> не происходит непосредственного описания методов взаимодействия с веб-сервисом. Они описываются ранее, в блоке <binding>, а в блоке <port> только дается ссылка на описанный метод связи binding.

9.2.5 Стандарт WSDL 2.0

В 2004 году был принят черновой вариант стандарта WSDL 2.0. WSDL 1.2 была переименована WSDL 2.0 из-за существенных отличий от WSDL 1.1. Не смотря на то, что до сих пор нет финальной спецификации данного стандарта, консорциум W3C в 2007 году рекомендовал использовать именно тот формат WSDL документов, который описан в стандарте WSDL 2.0.

В качестве основных изменений в стандарте WSDL 2.0 можно отметить следующее:

- многие блоки WSDL-документа были переименованы в соответствии с устоявшейся терминологией разработчиков распределенных приложений: Port превратился в Endpoint, PortType – в Interface;
- исчез блок Message, слившись с блоком Types (который стал обязательным);
- при описании методов связывания в блоке binding появилась возможность указывать любой метод HTTP запросов (GET, POST, PUT, DELETE) посредством атрибута whttp:method. Это сделало стандарт WSDL гораздо ближе к идеологии взаимодействия посредством веб.

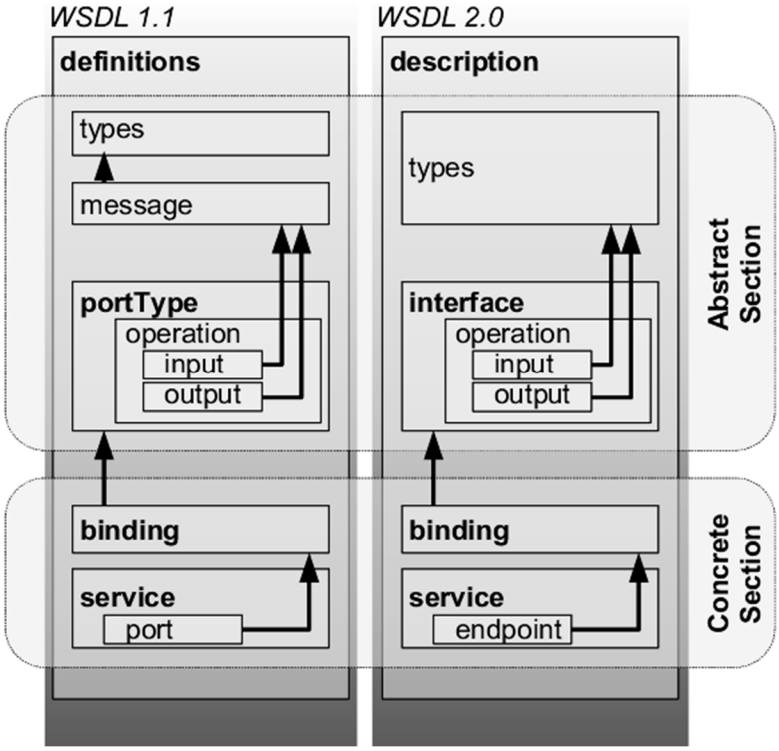

Рис. 35. Сравнение блоков WSDL 1.1 и WSDL 2.0

Однако поддержка WSDL 2.0 до сих пор является достаточно бедной в смысле программного обеспечения для разработки веб-сервисов.

9.3 Стандарт SOAP

Стандарт SOAP обеспечивает взаимодействие между веб-сервисами. Сообщением SOAP называют одностороннюю передачу информации от источника к приемнику между узлами SOAP. Сообщения SOAP являются основным строительным блоком, обеспечивающим возможности более сложных шаблонов взаимодействия: запрос/ответ, «диалоговый» режим и т.п.

Сообщение SOAP состоит из 3-х частей: *конверта*, содержащего *заголовок* и тело *сообщения*. В *теле* содержится XML-блок с информацией, которая должна быть доставлена конечному адресату. *Заголовок* – это не обязательный элемент SOAP-сообщения, при помощи которого можно передавать данные, не являющиеся собственно основной рабочей нагрузкой (к примеру: директивы и/или информацию о контексте, необходимые для обработки сообщения). SOAP-сообщение способно следовать по маршруту, содержащему несколько узлов, каждый из которых может вносить в него изменения или как-то еще его обрабатывать. Статус этих изменений отражается в блоках заголовка сообщения. Оба этих раздела содержатся внутри *конверта*.

```
<?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:Body>
</soap:Envelope>
```

Рис. 36. Шаблон SOAP-сообщения

В заголовке SOAP-сообщения мы имеем возможность ввести новые элементы сообщения, не предусмотренные стандартом SOAP. Эти элементы играют утилитарную роль по отношению к основному сообщению, содержащемуся в теле SOAP. Например, мы можем передать номер транзакции, в рамках которой пришло то или иное сообщения, передать информацию, необходимую для авторизации пользователя и др.

```
<soap:Header>
  <trans:Transaction
  xmlns:trans="http://www.host.com/namespaces/space/"
  soap:mustUnderstand="1">
  12
  </trans:Transaction>
</soap:Header>
```

Рис. 37. Пример заголовка SOAP-сообщения

У каждого элемента, введенного в заголовок SOAP-сообщения, мы можем указать значения следующих атрибутов:

– *mustUnderstand* – если значение данного атрибута равно 1, получатель обязан обрабатывать этот элемент заголовка. Если он не умеет этого делать, то он обязан отбросить полученное сообщение;
– *actor* – указывает название конкретного приложения-получателя если SOAP-сообщение проходит цепочку приложений при обработке.

В теле SOAP-сообщения производится передача сообщения по формату, определенному в блоках `<porttype>` и `<message>` WSDL-документа. Имя основного блока, находящегося в теле SOAP-сообщения соответствует имени сообщения, которое определено в интерфейсе веб-сервиса.

Запрос

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetails xmlns="http://warehouse.example.com/ws">
       <productID>12345</productID>
     </getProductDetails>
   </soap:Body>
 </soap:Envelope>
```

Ответ

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetailsResponse
xmlns="http://warehouse.example.com/ws">
       <getProductDetailsResult>
         <productID>12345</productID>
         <productName>Роутер NoName</productName>
         <description>Роутер NoName, WiFi, LAN</description>
         <price>550</price>
         <inStock>true</inStock>
       </getProductDetailsResult>
     </getProductDetailsResponse>
   </soap:Body>
 </soap:Envelope>
```

Рис. 38. Пример запроса и ответа посредством SOAP-сообщений

Как уже было сказано, стандарт SOAP обеспечивает одностороннюю передачу сообщений. Соответственно, для того чтобы получить ответ от сервера, которому было передано SOAP сообщение, может потребоваться чтобы он инициировал процесс передачи сообщение и установил соединение с клиентом. Это может вызвать значительные затруднения в современных условиях организации связи в сети Интернет, т.к. большинство клиентов не имеют выделенных статических IP-адресов. Даже наоборот, чаще всего входящие соединения блокируются сетевыми брандмауэрами.

Данная проблема решена в рамках стандартного связывания протокола SOAP и протокола HTTP, реализующего паттерн поведения «запрос-ответ» (анг. *SOAP HTTP Binding*).

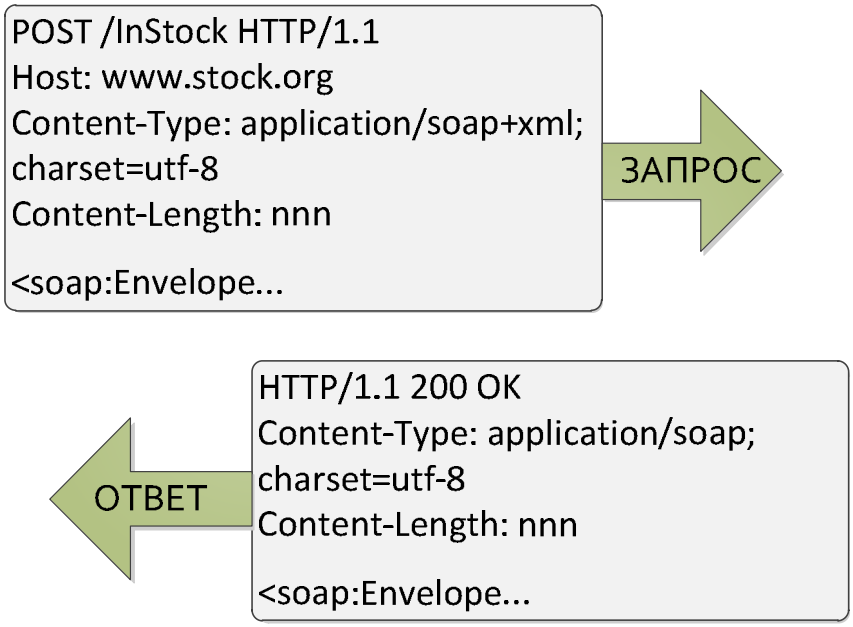

Рис. 39. Пример реализации паттерна взаимодействия «запрос-ответ» посредством HTTP-связывания

Заголовок «Content-Type» для сообщений HTTP-запроса и HTTP-ответа request устанавливается в значение text/xml (application/soap+xml в спецификации SOAP 1.2). HTTP-запрос должен использовать POST (начиная со спецификации SOAP 1.2 можно использовать GET). HTTP-ответ должен использовать статусный код 200 если обработка SOAP-сообщения прошла нормально, или же 500 если в теле сообщения содержится ошибка SOAP.

9.4 Второе поколение стандартов веб-сервисов

После появления технологии веб-сервисов в начале 2000-х годов, многие разработчики ощутили, что отсутствие стандартизации в наиболее важных областях построения распределенных вычислительных систем приводят к несовместимости разрабатываемых решений. Например, отсутствие единых протоколов авторизации и аутентификации пользователей приводило к тому, что каждому разработчику приходилось самостоятельно «придумывать велосипед» в этой области. Поэтому, когда возникала необходимость взаимодействия с внешней сервис-ориентированной системой приходилось тратить большое количество усилий на «скрещивание велосипедов», чтобы обеспечить стабильную работу объединенного решения.

Для решения этих проблем, множество коммерческих и некоммерческих организаций объединили свои усилия в рамках различных консорциумов для разработки нового поколения стандартов веб-сервисов (WS-* стандартов). К сожалению, это дало не совсем тот результат, на который рассчитывали с само-

го начала: каждый консорциум норовил разработать свой пакет стандартов, что привело к большому количеству различных стандартов, направленных на решение одних и тех же задач.

В рамках данной книги мы рассмотрим следующие, наиболее признанные и распространенные стандарты веб-сервисов:

– WS-Security – обеспечение безопасности веб-сервисов;
– WS-Addressing – маршрутизация и адресация SOAP-сообщений;
– WSRF, WS-Notification – работа с состоянием веб-сервисов.

9.4.1 Безопасность веб-сервисов и WS-Security

Стандарты веб-сервисов первого поколения не подразумевали обеспечения какой-либо безопасности при взаимодействии веб-сервисов. Разработчикам приходилось самостоятельно решать проблемы аутентификации и авторизации пользователей, разграничения прав доступа, передачи конфиденциальной информации и подписи сообщений. Это привело к тому, что практическое применение веб-сервисов в сфере бизнеса было ограничено.

Корпорации IBM и Microsoft совместно разработали семейство стандартов GXA – Global XML Web Services Architecture. В рамках данной архитектуры был предложен стандарт обеспечения безопасность веб-сервисов WS-Security. На схеме изображен стек протоколов обеспечения безопасности в GXA.

Рис. 40. Стек протоколов безопасности веб-сервисов на основе WS-Security

WS-Security является базой для других технологий в области безопасности веб-сервисов. В рамках данного стандарта описываются процессы обеспечения целостности (неизменности), конфиденциальности и неподдельности авторства (аутентичность) SOAP-сообщения, передаваемого в рамках уже установленных сессий, контекста и политики безопасности. В спецификации не говорится о

том, как формировать защищенное соединение, производить аутентификацию сторон, обмен ключами и обеспечивать другие аспекты безопасного общения. Это задачи других спецификаций GXA. Кроме того, в ней задается *только* рамочная *архитектура* – для производства реальных операций она полагается на уже имеющиеся технологии вроде PKI, Kerberos и SSL.

Таким образом, стандарт WS-Security ориентирован на комплексное решение задач безопасности при взаимодействии веб-сервисов, обеспечивая определение основных методов идентификации пользователя, цифровые подписи, шифрование.

WS-Security обеспечивает перемещение задач идентификации и авторизации в область обмена SOAP сообщениями. Чтобы обеспечить необходимый уровень безопасности SOAP-сообщения, информация, содержащаяся в SOAP-сообщении должна обеспечивать следующее:

- идентификацию категорий пользователей, связанных с сообщением;
- доказательство того, что категории пользователей имеют правильный набор прав доступа;
- доказательство того, что сообщение не изменялось.

WS-Security дает возможность применять маркеры безопасности (Security Tokens) при работе с SOAP сообщениями. Этот подход очень сильно напоминает «скидочные карточки», которые обеспечивают возможность получать товары и услуги и получать скидки в магазинах. Используя такие маркеры безопасности, SOAP сообщение может переправить следующую информацию [55]:

- *идентификацию вызывающего*: Я User Vasya Pupkin;
- *принадлежность к группе*: Я разработчик PupkinSite.com;
- *подтверждение прав*: поскольку я разработчик PupkinSite.com, я могу создавать базы данных и добавлять веб-приложения в сервера PupkinSite.com.

Чтобы создать сообщение, которое может создать новую базу данных на серверах *PupkinSite.com*, приложению придется запрашивать некоторые маркеры доступа. Эти маркеры доступа могут быть предоставлены в виде пары «имя пользователя/пароль», при помощи специальной смарт-карты, содержащей закрытый ключ шифрования или какого-либо другого метода. Когда *Vasya Pupkin* принимает решение создавать новую базу данных в *PupkinSite.com*, среда идет в Сервис присваивания удостоверений (Ticket Granting Service) и запрашивает Удостоверение сервиса (Service Ticket), который показывает, что *Vasya Pupkin* имеет право на создание новой базы данных на *PupkinSite.com*. Среда берет это

Удостоверение сервиса и представляет его серверу базы данных в *PupkinSite.com*. Этот сервер проверяет достоверность удостоверения и затем позволяет *Vasya Pupkin* создавать новый процесс.

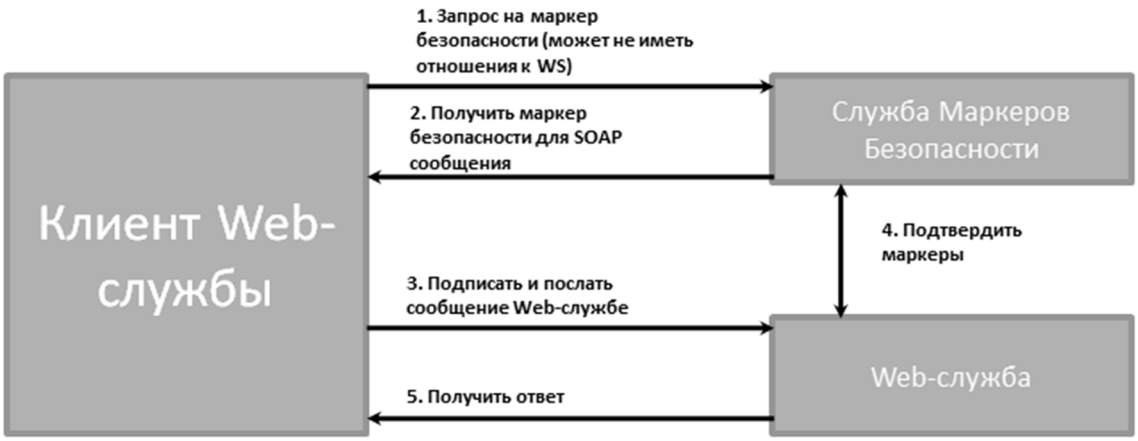

Рис. 41. Процедура авторизации пользователя на основе WS-Security

9.4.2 Аутентификация пользователя посредством WS-Security

Стандарт WS-Security предусматривает множество различных способов проверки достоверности пользователя. Из этого бесчисленного множества спецификация выделяет три метода:

– <*wsse:UsernameToken*> – аутентификация пользователя посредством пары «Имя пользователя/пароль»;
– <*wsse:X509v3*> – аутентификация посредством сертификата X.509v3;
– *Kerberos* – аутентификация посредством протокола Kerberos (Kerberos Domain Controller) (используется в Windows2000, Red Hat Linux и т.п.).

```
<soap:Envelope>
 <soap:Header>
    <wsse:Security soap:mustUnderstand="1">
       <wsse:UsernameToken>
          <wsse:Username>myName</wsse:Username>
          <wsse:Password Type="wsse:PasswordText">myPassword
           </wsse:Password>
       </wsse:UsernameToken>
   </wsse:Security>
 </soap:Header>
 ...
</soap:Envelope>
```

Рис. 42. Внедрение имени пользователя и пароля в заголовок SOAP-сообщения в виде открытого текста

В приведенном примере вы можете заметить префикс «wsse» у некоторых блоков, входящих в заголовок или тело SOAP-сообщения. Этот префикс означает, что формат данного XML-элемента определен в спецификации WS-Security.

Один из наиболее распространенных способов передачи удостоверения пользователя – использование комбинации имени пользователя и пароля. Для передачи удостоверения пользователя таким способом, в WS-Security определен элемент UsernameToken. Этот фрагмент использует два других типа: Username и Password. Эти два типа, представляют собой строки, которые, если необходимо, могут содержать дополнительные атрибуты.

Пароль может передаваться как простой текст или в цифровом формате. Передача имени пользователя и пароля в открытом формате обычно применяется в том случае, если обмен SOAP-сообщениями ведется поверх установленного защищенного соединения. При передаче пароля в цифровом виде производится его хеширование на основе случайного ключа и информации о времени формирования сообщения. Такой алгоритм обеспечивает возможность производить авторизацию пользователя по имени и паролю даже в том случае, если обмен сообщениями происходит по открытому каналу.

```
<wsse:UsernameToken>
    <wsse:Username>scott</wsse:Username>
    <wsse:Password Type="wsse:PasswordDigest">
        KE6QugOpkPyT3Eo0SEgT30W4Keg=</wsse:Password>
    <wsse:Nonce>5uW4ABku/m6/S5rnE+L7vg==</wsse:Nonce>
    <wsu:Created xmlns:wsu=
        "http://schemas.xmlsoap.org/ws/2002/07/utility">
            2002-08-19T00:44:02Z
    </wsu:Created>
</wsse:UsernameToken>
```

Рис. 43. Передача пароля в виде цифрового хэша

Другим вариантом авторизации пользователей является аутентификация на основе сертификата X.509. Сертификат X.509 позволяет однозначно определить, кем конкретно является пользователь системы. Использование сертификата по-своему может способствовать осуществлению очень простых атак воспроизведения. В результате, нелишним будет заставлять отправителя сообщения также подписывать его с помощью секретного ключа. Таким образом, когда сообщение получает ключ для дешифровки, вы будете знать, что это действительно пользователь.

Когда сообщение посылает сертификат X.509, оно передаст открытую версию сертификата в маркер WS-Security BinarySecurityToken. Сам сертификат получает отправленное сообщение как шифрованные base64 данные. Для обеспечения безопасности при использовании сертификата надо прибегнуть к дополнительным средствам обеспечения безопасности: подпись сообщения секретным ключом сертификата и добавлению `wsu:Timestamp` для определения времени жизни сообщения.

В заголовке WS-Security аутентификация пользователя посредством сертификата X.509 будет выглядеть примерно так:

```
<wsse:BinarySecurityToken
ValueType="wsse:X509v3"
EncodingType="wsse:Base64Binary"
Id="SecurityToken-f49bd662-59a0-401a-ab23-1aa12764184f">
   MIIHdjCCBCCAWqgAwIBAgIBGzANBgkqh-
kiG9w0BAQQFADBHMQ0wCwYDVQQKEwRMRwwGgYDV
   QQLExNDYWViZW-
FucyBkZXZlbG9wZXJzMRgwFgYDVQQLEw9jYWViZWFucy5uZXQucnUwHhcN
   MDcxMjAxMDAwMDAwWhcNMDgwMTMxMjM1OTU5Wj...
</wsse:BinarySecurityToken>
```

Рис. 44. Аутентификация на основе сертификата X.509

9.4.3 Подпись сообщения

Подпись сообщения позволяет получателю SOAP-сообщения удостовериться в том, что подписанные элементы не были изменены по пути передачи сообщения. При этом не надо забывать, что подпись сама по себе не может защитить сообщение от просмотра его содержимого третьими лицами.

Непосредственно за подпись сообщения отвечает спецификация XML Signature. WS-Security просто описывает, как использовать данный формат для подтверждения того, что SOAP-сообщение не было изменено по пути следования.

В зависимости от выбранного метода аутентификации, при подписи сообщения может быть использована следующая информация:

– UsernameToken – *пароль пользователя*;
– X.509 – *секретный ключ*;
– Kerberos – *сеансовый ключ*.

```
<soap:Envelope>
 <soap:Header>
    <wsse:Security soap:mustUnderstand="1">
    ...
      <ds:Signature>
        <ds:Signedlnfo>
                  ...
        </ds:Signedlnfo>
        <ds:SignatureValue>
            Hp1ZkmFZ/2kQLXDJbchm5gK...
        </ds:SignatureValue>
     <ds:Keylnfo>
        <wsse:SecurityTokenReference>
           <wsse:Reference URI=" #X509Token"/>
         </wsse:SecurityTokenReference>
     </ds:Keylnfo>
   </ds:Signature>
    </wsse:Security>
 </soap:Header>
 ...
</soap:Envelope>
```

Рис. 45. Подпись сообщения на основе сертификата X.509

9.4.4 Шифрование

Аутентификация и подпись сообщения – это не всегда достаточная мера обеспечения безопасности, особенно при передаче конфиденциальной информации. Как и с подписью сообщений, разработчики спецификации WS-Security приняли уже существующий стандарт, который хорошо выполняет работу, связанную с шифрованием. За шифрование отвечает стандарт XML Encryption.

```
<soap:Envelope>
...
<soap:Body>
      <xenc:EncryptedData
      Id="EncryptedContent-f6f50b24-3458-41d3-aac4-390f476f2e51"
      Type="http://www.w3.org/2001/04/xmlenc#Content">
         <xenc:EncryptionMethod Algorithm=
         "http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
         <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Symmetric Key</KeyName>
         </KeyInfo>
         <xenc:CipherData>
            <xenc:CipherValue>
          InmSSXQcBV5UiT...  Y7RVZQqnPpZYMg==
          </xenc:CipherValue>
         </xenc:CipherData>
      </xenc:EncryptedData>
...
</soap:Envelope>
```

Рис. 46. Шифрование сообщения на основе XML Encryption

Как можно увидеть из представленного примера, все содержимое тела SOAP-сообщения шифруется и заменяется на блок `<xenc:EncryptedData>`, кото-

рый может быть прочитан только при наличии секретной авторизационной информации.

9.5 Адресация и WS-Addressing

В стандартах первого поколения полный адрес веб-сервиса содержался в WSDL-описании, в блоке `<port>`. Это может доставлять значительные неудобства, т.к. при изменении адреса сервиса приходится редактировать WSDL-файл целиком. А при обмене сообщениями SOAP, адресация возложена на транспортный протокол (при связывании с HTTP) и не может быть изменена непосредственно в SOAP-сообщении.

В стандарте WS-Addressing предусматривается введение полей `<wsa:To>` и `<wsa:Action>` в заголовок SOAP-сообщения, определяющих URI приемника сообщения и соответствующее действие:

В стандарте первого поколения подразумевается, что ответное сообщение передается по уже открытому HTTP каналу в соответствии с правилами HTTP-связывания. При этом нет стандартной поддержки асинхронной коммуникации между веб-сервисами. Стандарт WS-Addressing вводит следующие поля: `<MessageID>`, `<From>` (адрес отправителя), `<ReplyTo>` (адрес получения ответа), `<FaultTo>` (адрес извещения об ошибках), `<RelatedTo>` (позволяет выстраивать сообщения в цепочки).

Рис. 47. Адресация посредством WS-Addressing

Стандарт WS-Addressing обеспечивает расширенную адресацию конечных точек в WSDL-файле. Так как WSDL не поддерживает расширение элемента <Service>, в WS-Addressing определен элемент <EndpointReference>, который может быть использован в WSDL.

<EndpointReference> расширяет элемент <Service> добавляя поля ReferenceProperties и Policy. Поля Address, ServiceName и PortType уже включены в элемент <Service>.

EndpointReference:

```
<wsa:EndpointReference>
     <wsa:Address />
     <wsa:ReferenceProperties />
     <wsa:ServiceName PortName=””/>
     <wsa:PortType />
     <wsa:Policy />
</wsa:EndpointReference>
```

WSDL Service:

```
<wsdl:service name="MyMathService">
      <wsdl:port binding="impl:MyMathSoapBinding“ name="MyMath">
           <wsdlsoap:address
           location="http://supercomputer.susu.ru/MyMath"/>
      </wsdl:port>
 </wsdl:service>
```

Рис. 48. Сравнение EndpointReference и блока WSDL Service

Рассмотрим основные поля, входящие в состав адреса конечной точки по стандарту WS-Addressing:

- <wsa:Address /> – URI, который идентифицирует конечную точку;
- <wsa:ReferenceProperties /> – может содержать отдельные свойства, необходимые для идентификации лица или ресурса (например, имя файла, с которым должно производиться определенное действие или идентификатор корзины покупателя;
- <wsa:PortType />; <wsa:ServiceName /> – используются по аналогии с блоками PortType и ServiceName WSDL;

Также существует ряд необязательных полей (policy, reference parameters, selected port type, service-port).

Рассмотрим пример, каким образом может быть сформировано SOAP-сообщение на основе блока <EndpointReference> на примере веб-сервиса интернет-магазина. На рисунке представлен пример информации, хранящейся в ссылке на конечную точку. Приложение, работающее с интернет магазином, сохранило не только адрес веб-сервиса, который необходимо вызывать для по-

лучения доступа к магазину, но также и уникальную идентификационную информацию, которая позволяет узнать, от какого покупателя поступил запрос (блок <fabrikam:CustomerKey>) и уникальный идентификатор его корзины (<fabrikam:ShoppingCart>). Эти данные хранятся в блоке дополнительных свойств ссылки <wsa:ReferenceProperties />.

Рис. 49. Формирование SOAP-сообщения на основе <EndpointReference>

При формировании SOAP-сообщения, данные свойства переходят в его заголовок и передаются веб-сервису, который знает, каким образом обрабатывается эта информация. Таким образом, стандарт WS-Addressing не только обеспечивает независимые методы адресации веб-сервисов, но и возможность передачи и хранения дополнительной информации в заголовках SOAP-сообщений.

9.6 Состояние веб-сервисов и WSRF

Изначально, использование веб-сервисов не подразумевало существования «состояния» (этот вопрос был подробно рассмотрен в разделе 8.4 «Подход СОА»). Типовой сценарий использования веб-сервиса подразумевал шаблон «запрос – ответ – отключение». При этом каждый следующий запрос недолжен зависеть от результатов предыдущего запроса.

Но для многих приложений, как коммерческих, так и научных, сохранение информации о состоянии было существенным требованием успешного функционирования. Например, для разработки грид-систем не получилось применить «чистые» веб-сервисы, т.к. они не обладали возможностью работа с состоянием. В течение довольно продолжительного времени, отсутствие стандартных технологий обработки состояния привело к появлению множества несовмести-

мых друг с другом вариантов «симуляции» работы с состоянием посредством веб-сервисов.

Спецификация Web Services Resource Framework (WFRF) является попыткой решить указанную архитектурную проблему с помощью введения понятия «состояние» в веб-сервисы, превратив их в веб-ресурсы, и указав механизмы использования этого понятия.

Спецификация WSRF была предложена в 2004-м году, утверждена в качестве стандарта OASIS в 2006-м году. WSRF – это ряд спецификаций, которые определяют стандартные способы запроса значений свойств или способы указания того, что эти свойства должны быть изменены. Также WSRF определяет стандартные способы разрешения всех различных проблемных вопросов работы с WS-ресурсами.

Спецификация WSRF включает следующие стандарты:

– *WS-Resource specification* – описание WS-ресурсов;
– *WS-ResourceProperties (WSRF-RP)* – описание свойств WS-ресурсов;
– *WS-ResourceLifetime (WSRF-RL)* – описание управления временем жизни (создание и уничтожение) WS-ресурсов;
– *WS-ServiceGroup (WSRF-SG)* – работа с группами ресурсов;
– *WS-BaseFaults (WSRF-BF)* – описание основных ошибок, которые могут возникнуть при работе с WS-ресурсами.

В спецификации *WS-ResourceLifetime (WSRF-RL)* определяется, когда WS-ресурс должен прекратить свою активность или быть явно уничтожен, когда пропадает необходимость его существования. Спецификация *WS-ServiceGroup (WSRF-SG)* определяет способ создания набора веб-сервисов, такого, например, как регистр имеющихся сервисов. Спецификация *WS-Base Faults (WSRF-BF)* определяет стандартный способ фиксации ошибок в WSRF-приложении. В этом списке нет спецификации, определяющей как все это должно работать совместно. Эту функцию выполняет (почти полностью) документ Modeling stateful resources with Web services (Моделирование ресурсов с состоянием посредством веб-сервисов) [22], в котором объясняется общая концепция этих спецификаций и их связь друг с другом.

В соответствии со спецификацией, *WS-ресурс* является объединением веб-сервиса и ресурса с состоянием, на который веб-сервис может воздействовать.

9.6.1 Процесс создания WS-ресурсов

Состояние объекта можно определить через значения его различных свойств. Мы можем представить наш ресурс с состоянием в виде XML-

документа, в котором содержатся его свойства. Такой документ называется *Resource properties document*. В качестве примера, сформируем список свойств такого ресурса, как космический спутник. В качестве свойств ресурса определим такие его характеристики как широта, долгота, углы наклона, высота над земной поверхностью, а также текущий вид из объектива фотокамеры, установленной на данном спутнике.

```
<satProp:GenericSatelliteProperties
xmlns:satProp="http://example.com/satellite">
 <satProp:latitude>30.3</satProp:latitude>
 <satProp:longitude>223.2</satProp:latitude>
 <satProp:altitude>47700</satProp:altitude>
 <satProp:pitch>49</satProp:pitch>
 <satProp:yaw>0</satProp:yaw>
 <satProp:roll>32</satProp:roll>
 <satProp:focalLength>21999992</satProp:focalLength>
 <satProp:currentView>
       http://example.com/satellite/2239992333.zip
 </satProp:currentView>
</satProp:GenericSatelliteProperties>
```

Рис. 50. Свойства WS-ресурса «космический спутник»

К этому моменту мы создали модель нашего ресурса с состоянием (спутника), но чтобы завершить создание WS-ресурса, мы должны связать его с нашим сервисом, используя WSDL-файл. Для добавления информации о свойствах нашего ресурса, сформируем типы данных, которые будут представлять наш WS-ресурс в WSDL-файле.

```
<definitions name="Satellite" ...>
 ...
 <types>
       <xsd:schema targetNamespace="http://example.com/satellite"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
             ...
             <xsd:element name="latitude" type="xsd:float" />
             <xsd:element name="longitude" type="xsd:float" />
             <xsd:element name="altitude" type="xsd:float" />
             ...
             <xsd:element name="GenericSatelliteProperties">
             <xsd:complexType>
                   <xsd:sequence>
                   <xsd:element ref="latitude" minOccurs="1"
                   maxOccurs="1"/>
                   <xsd:element ref="longitude" minOccurs="1"
                   maxOccurs="1"/>
                   ...
                   </xsd:sequence>
             </xsd:complexType>
             </xsd:element>
       </xsd:schema>
 </types>
       <portType name="SatellitePortType"
             wsrp:ResourceProperties=
             "tns:GenericSatelliteProperties">
       </portType>
</definitions>
```

Рис. 51. Свойства WS-ресурса в WSDL-файле

Мы начали с добавления базовых элементов веб-сервиса, элемента `service` и элемента `binding`, который ассоциирует `service` с элементом `portType`. Сам

элемент `portType` пока что не содержит в себе никаких операций, самой главной составляющей в нем является атрибут `wsrp:ResourceProperties`. Этот атрибут говорит о том, что любая операция, которую выполняет наш веб-сервис, совершается над определенным типом ресурса с состоянием, так, как это определено элементом `GenericSatelliteProperties`. Элемент `GenericSatelliteProperties` определен в элементе `schema`. Объединение этого ресурса с состоянием и нашего веб-сервиса и является требуемым WS-ресурсом.

Теперь добавим несколько операций по работе с ресурсом в базовые элементы WSDL-описания нашего веб-сервиса.

```
...
 <types>
        ...
        <xsd:element name="createSatellite">
              <xsd:complexType/>
        </xsd:element>
        <xsd:element name="createSatelliteResponse"> <xsd:complexType>
              <xsd:sequence>
              <xsd:element ref="wsa:EndpointReference"/>
              </xsd:sequence>
        </xsd:complexType> </xsd:element>
        ...
</types>
<message name="CreateSatelliteRequest">
 <part name="request" element="tns:createSatellite">
</message>
<message name="CreateSatelliteResponse">
 <part name="response" element="tns:createSatelliteResponse"/>
</message>
<portType name="SatellitePortType" wsrp:ResourceProperties=
              "tns:GenericSatelliteProperties">
 <operation name="createSatellite">
        <input message="tns:CreateSatelliteRequest"
              wsa:Action="http://example.com/CreateSatellite" />
        <output message="tns:CreateSatelliteResponse"
              wsa:Action="http://example.com/CreateSatelliteResponse"/>
 </operation>
</portType>
```

Рис. 52. Описание операций по работе с WS-ресурсом

Нужно обратить внимание на одну вещь. Вместо того чтобы возвращать простое значение, сервис возвращает *EndpointReference, где содержится ссылка на вновь созданный WS-ресурс*. Посмотрим, как это используется в SOAP-сообщении.

Запрос:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
        <createSatellite xmlns="http://example.com/satellite"/>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ответ:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.
org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
      <wsa:EndpointReference
      xmlns:wsa="http://www.w3.org/2005/02/addressing"
      xmlns:sat="http://example.org/satelliteSystem">
           <wsa:Address>http://example.com/satellite</wsa:Address>
           <wsa:ReferenceProperties>
                <sat:SatelliteId>SAT9928</sat:SatelliteId>
           </wsa:ReferenceProperties>
      </wsa:EndpointReference>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Рис. 53. Запрос на создание ресурса

У нас еще нет фактического объекта, поэтому сам запрос направляется согласно URI, записанному в WSDL-файле, и мы определили содержание запроса как простой элемент `createSatellite`. После того, как мы послали запрос на создание нового спутника, сервер создает ссылку на новый WS-ресурс и отсылает ее обратно в форме *EndpointReference*.

Заметим, что элемент *Address* в *EndpointReference* указывает на тот же самый URI, который мы поместили в WSDL-файл, поэтому информация отправляется, как и раньше, по прежнему адресу; только информации стало больше. Необходимо также отметить, что мы имеем дело не с обычной конечной ссылкой. В элементе *EndpointReference* указан идентификатор, который обязательно должен использоваться при идентификации WS-ресурса. В связи с этим, этот элемент можно рассматривать как действительно квалифицированную конечную ссылку на WS-ресурс.

Элемент wsa:Action не является частью созданной ссылки на конечную точку; он изменяется в зависимости от того, что мы пытаемся сделать. В рассматриваемом случае мы используем действие GetResourceProperty. Элемент wsa:To выбирает значение по адресу wsa:Address (из ссылки на крайнюю точку), и любые значения wsa:ReferencsProperty непосредственно переносятся в заголовок (Header).

Вы заметили, что мы не обсуждали значение SatelliteId. Это было сделано специально. Любая, содержащаяся в ссылке на крайнюю точку информация, относящаяся к идентификации конкретного WS-ресурса, должна игнорироваться вашим приложением, вы просто передаете ее при отправлении сообщений. В соответствии со спецификацией, считается некорректным даже попытка интерпретации значения SatelliteId. Безусловным является предположение, что оно

передается как «черный ящик», ведущий независимую и недоступную для наблюдений жизнь.

```
    <SOAP-ENV:Envelope>
    <SOAP-ENV:Header>
     <wsa:Action>
          http://docs.oasis-open.org/wsrf/2004/06/WS-
ResourceProperties/GetResourceProperty
     </wsa:Action>
     <wsa:To SOAP-ENV:mustUnderstand="1">
          http://example.com/satellite
     </wsa:To>
     <sat:SatelliteId>SAT9928</sat:SatelliteId>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
     <wsrp:GetResourceProperty
          xmlns:satProp="http://example.com/satellite">
          satProp:altitude
     </wsrp:GetResourceProperty>
    </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Рис. 54. Запрос на получение информации о состоянии ресурса

Таким образом, мы рассмотрели основные, на сегодняшний день, стандарты технологии веб-сервисов, которые применяются как в коммерческих так и в научных распределенных вычислительных системах. Одной из крупнейших областей, где стандарты веб-сервисов второго поколения нашли свое применение, стала область грид-технологий, о которой мы поговорим в главе 11.