

**Взаимодействия, сохраняющие состояние в Web-службах**  
**Сравнение подходов WS-Context и WS-Resource Framework**  
Марк Литтл, Джим Вебер, Савва Парастатидис

**Stateful Interaction in Web Services**  
**A comparison of WS-Context and WS-Resource Framework**  
Mark Little; Jim Webber; Savas Parastatidis

<http://webservices.sys-con.com/read/44675.htm>

CFID=52129&CFTOKEN=E33323B...

*В июле 2003г. консорциум производителей Web-служб выпустил релиз "Платформа композитного приложения Web-служб" (Web Services Composite Application Framework WS-CAF). Этот документ содержит три спецификации, которые предоставляют средства для надёжного построения больших композитных приложений из отдельных Web-служб. Краеугольным камнем этого комплекта является спецификация WS-Context, определяющая механизм управления сохранением состояния при взаимодействиях между Web-службами. Документ WS-CAF затем был передан в комитет OASIS, где в настоящее время рассматривается вопрос стандартизации предложенной платформы.*

В январе 2004 г. группа специалистов из академических и промышленных структур грид-сообщества опубликовала первую часть комплекта спецификаций "Платформа WSRF" (Web Services Resource Framework). Эти спецификации будут использоваться для управления сохраняющими состояние взаимодействиями между потребителями и ресурсами, обслуживаемыми Web-службами.

Ясно, что имеет место некоторое перекрытие между подходами WS-Context и WSRF, поскольку они оба поддерживают выполнение сохраняющих состояние взаимодействий, которые происходят поверх модели взаимодействий, не сохраняющих состояние, принятой в концепции архитектуры Web-служб (Web Service Architecture - WSA). В данной работе исследуются различия подходов WS-Context и WSRF, при этом особое внимание уделено тому, как каждый из них облегчает использование в композитных приложениях, основанных на Web-службах, взаимодействий, сохраняющих состояние.

## Подход WS-Context

Композитное приложение представляет собой набор реализуемых с помощью Web-служб активностей, выполняемых в некой заданной последовательности. Способность устанавливать рамки произвольных единиц распределённой работы (известных как активности) является требованием в разнообразии аспектов распределённых приложений, например таких, как поток рабочих заданий, B2B взаимодействия, автоматизированные бизнес-процессы и другие. Будучи охвачены общей работой, её участники могут внутри активности недвусмысленно определять, находятся они или нет внутри той же самой активности.

Для того чтобы коррелировать работу служб, участвующих внутри одной и той же активности, необходимо распространять для них дополнительную информацию, известную как *контекст*. Контекст содержит такую информацию, как уникальный ID и позволяет сериям активностей разделять общий результат. Согласно спецификации WS-Context в заголовки SOAP-сообщений заносится контекстная информация, которая распространяется с помощью сообщений на уровне приложения. Этот контекст позволяет множеству участников коррелировать SOAP-сообщения при обменах, для того, чтобы создавать большую абстракцию, например, такую, как поток процесса, безопасные переговоры или другое объединение.

Хотя распространение контекста является фундаментальным требованием для многих распределённых систем, включая Web-службы, тип используемой информации о контексте, может варьироваться в зависимости от обстоятельств. Например, в системе обработки транзакций это может быть URI для координатора, в то время как для безопасной передачи данных это может быть открытый ключ шифрования отправителя. Соответственно WS-Context-подход был разработан как стандартизированное средство передачи контекстной информации Web-службам.

Спецификация WS-Context также определяет службу контекста, которой могут воспользоваться Web-службы при формировании композитных приложений. Так как при использовании контекста может потребоваться различная информация, которая должна быть передана, спецификация WS-Context предусматривает минимальный (но расширяемый) контекст, который позволяет службам регистрировать контекстные связи и настраивать контексты на базе активностей.

## Подход WS-Resource Framework

Комплект спецификаций WSRF был разработан в ответ на ряд предложений, появившихся после того, как сообщество Web-служб начало использовать OGSИ-спецификации (например, WS-GAF предложения). Представители IBM и Globus Alliance предложили спецификации WSRF в качестве "точки конвергенции" между Web- и грид-службами, и они были позиционированы как естественная эволюция спецификаций Open Grid Services Infrastructure (OGSI).

Подход WSRF следует той же самой концептуальной модели, базирующейся на разделении ресурсов, которую поддерживает OGSИ, но без переделки ниже лежащих спецификаций Web-служб. При этом в нём учтены многие из поступивших предложений, особенно в плане того, что касается факторизации, контекстуализации при моделировании сохраняющих состояние взаимодействий, чёткого разделения между понятиями *служба* и

*ресурс*, а также не модифицированного использования существующих технологий Web-служб.

Весь комплект спецификаций, предусмотренных подходом WSRF, пока ещё полностью не опубликован; доступны лишь спецификации, позволяющие описывать состояние ресурса, время жизни ресурса и уведомления, в то время как спецификации, касающиеся групп служб, обновления ссылок на ресурс и отказов, будут опубликованы позже. Поскольку в центре внимания данной работы находятся взаимодействия, сохраняющие состояние, мы сосредоточимся именно на этом аспекте.

Согласно концептуальной модели подхода WSRF, Web-служба рассматривается как не имеющая состояния сущность, "которая действует, предоставляет доступ или обрабатывает набор логических ресурсов (документов), обладающих состоянием, на основании отправляемых или получаемых ею сообщений". Такая модель предполагает явное экспонирование свойств (логических или физических) ресурсов через границы службы. Представление состояния этих экспонируемых ресурсов и аппарат, с помощью которого потребители могут непосредственно взаимодействовать с ними, и являются главной целью подхода WSRF.

### **Поддержка взаимодействий, сохраняющих состояния**

Несмотря на то, что цели, преследуемые подходами WSRF и WS-Context одинаковы, в их спецификациях предлагаются принципиально разные способы достижения этих целей (обработка ресурсов против обработки композиции служб). В частности, подходы разнятся в том, как они решают проблему сохранения состояния при взаимодействиях между службами, поскольку спецификации WSRF поддерживают подход, ориентированный *на ресурс*, который связан с данной службой, в то время как спецификации WS-Context обеспечивают подход, ориентированный *на контекст*, в котором происходит выполнение данной службы.

Для того чтобы продемонстрировать применение подхода WS-Context для поддержки сохраняющих состояние взаимодействий, мы проанализируем простейший случай его использования, когда сохраняющее состояние взаимодействие происходит между одной службой и одним клиентом.

Ранее мы обсудили, как WS-Context-подход определяет понятие активности, для которой контекст ограничен. Каждая активность гарантирует, что каждое взаимодействие, проходящее через *WS-Context-осведомляемую службу*, будет уникально и недвусмысленно привязано к этой активности через контекст. В рассматриваемом простом примере, контекст используется для идентификации потребителем конкретного взаимодействия, сохраняющего состояние, и для идентификации с помощью службы конкретного диалогового состояния.

Подход WS-Context предусматривает следующие этапы жизненного цикла контекста:

1. Чтобы инициировать активность, одна из служб запрашивает у WS-Context-службы новый контекст, для чего посылает ей *начальное* сообщение. Иницилирующая активность служба может установить лимит времени, отведенного для сессии, или может указать, что сессия будет продолжаться до тех пор, пока не поступит приказ о её завершении. Кроме того, в зависимости

- от требований приложения, контекст может быть автоматически создан службой в ответ на поступление первого запроса от потребителя.
2. Начальное действие возвратит начальное сообщение плюс контекст.
  3. Всякий раз, когда потребитель взаимодействует с WS-Context-осведомляемой службой, контекст распространяется через блок SOAP заголовка передаваемого сообщения. Служба, принимающая сообщение, должна уметь обращаться с состоянием любого конкретного контекста, который она запрашивает для того, чтобы коррелировать сообщения.
  4. Взаимодействие, сохраняющее состояние, может быть прекращено либо по исчерпанию лимита времени, установленного для сессии, либо при получении приказа службе контекста закончить активность.

Концепция WSRF-подхода выдвигает неявную контекстуализацию в качестве механизма для поддержки сохраняющих состояние взаимодействий между потребителями и ресурсами, экспонированными за границами службы (концепция не моделирует контекст как некую внешнюю сущность так, как это принято в концепции подхода WS-Context). Такие ресурсы, логические или физические, идентифицируются с помощью конструкций WS-Addressing-спецификаций. В дополнение к информации о крайних точках службы эти конструкции также содержат конкретную информацию о ресурсах, которая помещена в элемент `<wsa:ReferenceProperty/>` конструкции WS-Addressing.

Когда потребитель вступает во взаимодействие, сохраняющее состояние с идентифицированным ресурсом, ему необходимо включать содержимое элемента `<ReferenceProperties/>` в качестве заголовка в каждом SOAP-сообщении, посылаемом службе, идентифицированной элементом `<EndpointReference/>` той же самой конструкции WS-Addressing. Служба, получающая сообщение, будет использовать эту информацию для маршрутизации обращений к ресурсу (см. рис. 1).

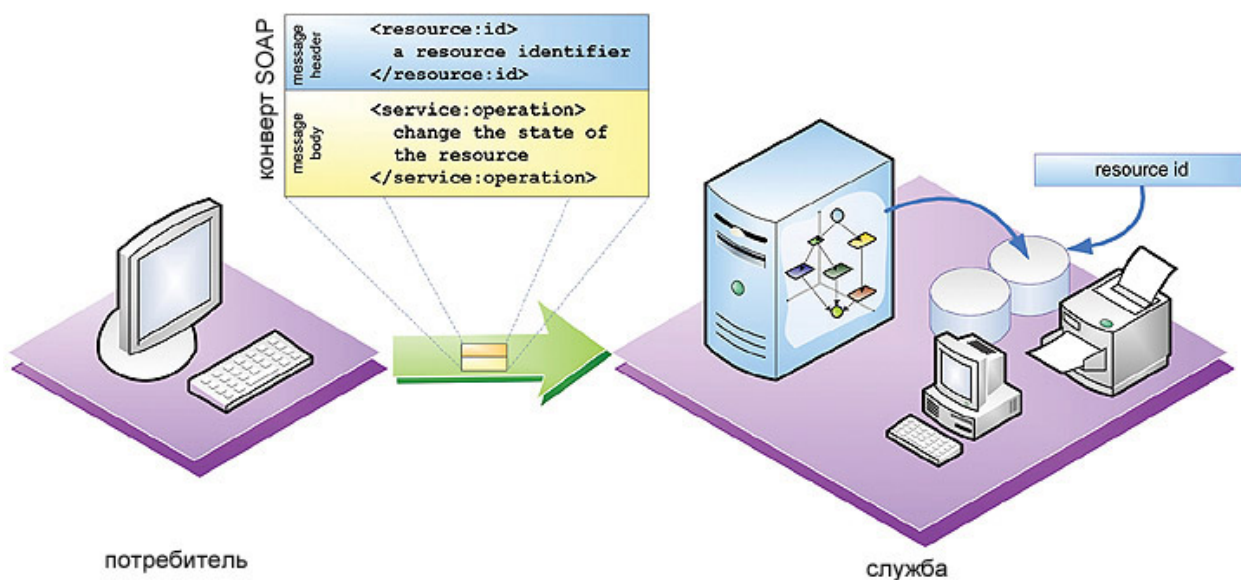


Рис. 1

Согласно толкованию подхода WSRF, конструкция WS-Addressing считается бессмысленной для её потребителей, и поэтому им не следует пытаться как-то перерабатывать эту специальную информацию о ресурсе. Информация о ресурсе

считается закрытым элементом описания службы и может использоваться только данной службой. Фактически, в спецификациях WSRF конструкции WS-Addressing используются в качестве сетевых указателей на ресурсы (См. листинг 1). Для этого каждому потребителю необходимо включать элемент `<example:DataSetId/>` в заголовок каждого SOAP-сообщения, требующего исполнения некой явно идентифицированной операции, которая должна быть выполнена на ресурсе (например, сообщение, требующее чтобы указанный набор данных был отсортирован или удалён). Этот элемент используется принимающей службой для идентификации и делегирования обращений к соответствующему внутреннему ресурсу.

### Листинг 1

#### Пример спецификации WS-Addressing с информацией о ресурсе

```
<wsa:EndpointReference xmlns:wsa="... "xmlns:example="...">
  <wsa:Address>http://www.example.com/webservice</wsa:Address>
  <wsa:PortType>example:ResourceOperations</wsa:PortType>
  <wsa:ReferenceProperties>
    <example:DataSetId>dataset15</example:DataSetId>
  </wsa:ReferenceProperties>
</wsa:EndpointReference>
```

В качестве сетевого указателя конструкция WS-Addressing, содержащая специальную информацию о ресурсе, служит той же цели, что конструкции *CORBA IOR*, *DCOM OBJREF*, *Java RMI URL* и т. д.; она идентифицирует ресурс по всей сети. Придерживаясь методологии существующих объектно-ориентированных технологий распределённого компьютеринга, подход WSRF переносит проблему идентификации ресурса с уровня приложения вниз (*на уровень промежуточного программного обеспечения*) и решает её в рамках стека Web-служб.

Требую, чтобы идентичность ресурса передавалась в качестве заголовка каждого SOAP-сообщения, подход WSRF моделирует взаимодействия, сохраняющие состояние, с помощью конкретных ресурсов, а не служб. Если учесть наличие дополнительных спецификаций, которые обеспечивают управление сроком жизни экспонированных ресурсов и механизмом обновления ссылок на эти ресурсы, то можно утверждать, что подход WSRF имеет много общего с концепциями моделей распределённых объектов.

#### Пример хранилища файлов

Чтобы представить проблемы, поднятые в предыдущих разделах, исследуем гипотетическую реализацию простого хранилища файлов, основанного на Web-службах. (Замечание: выбор в качестве объекта исследования хранилища файлов сделан потому, что оно простое, и вместе с тем является каноническим примером для демонстрации достоинств и недостатков как WSRF, так и WS-CAF внутри обоих сообществ). В этом примере реализация хранилища файлов упрощена; для ясности мы игнорируем такие вопросы, как техническая политика и безопасность, хотя в практической реализации оба эти свойства были бы критическими. Общий вид реализации хранилища файлов показан на рис. 2.

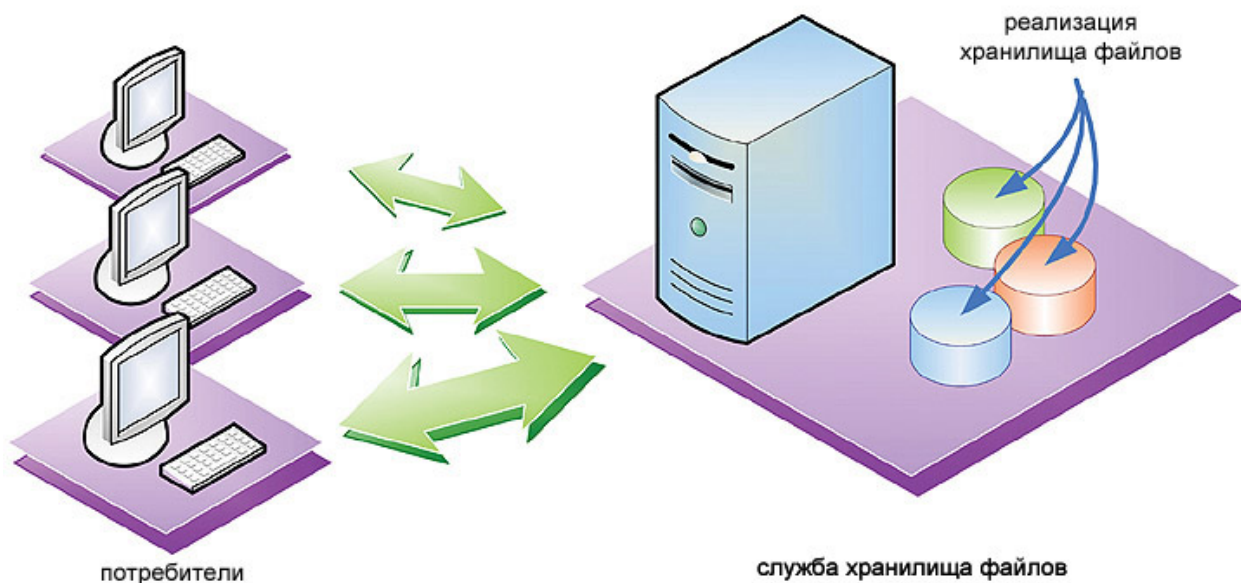


Рис. 2

При использовании подхода WSRF схема доступа к хранилищу файлов достаточно прямолинейна. Для конкретного ресурса получена WS-Addressing-ссылка на крайнюю точку (что осуществлено через некоторый вспомогательный механизм, подобный регистрации или фабрике). Эта ссылка на крайнюю точку (расширенная специфическими WSRF-метаданными) рассматривается как сетевой указатель для ресурса, ведомого (hosted) Web-службой. Полученная ссылка на крайнюю точку может использоваться и как адрес, по которому может быть отправлено сообщение, и как неявный контекст, для взаимодействия с внутренним ресурсом (для чего можно воспользоваться содержимым его элемента `<ReferenceProperties/>`).

В случае с хранилищем файлов, ID файла (идентификатор узла или некоторый другой дескриптор) может использоваться для того, чтобы предоставить метаданные, необходимые службе для маршрутизации обращения к одному и тому же ресурсу, при каждом сообщении, переданном службе. Всякий раз, когда клиент (см. рис. 3) посылает сообщение службе, её крайняя точка идентифицируется WS-Address-элементом, а метаданные, связанные с ресурсом, используются для того, чтобы помочь службе найти маршрут к правильному внутреннему ресурсу.

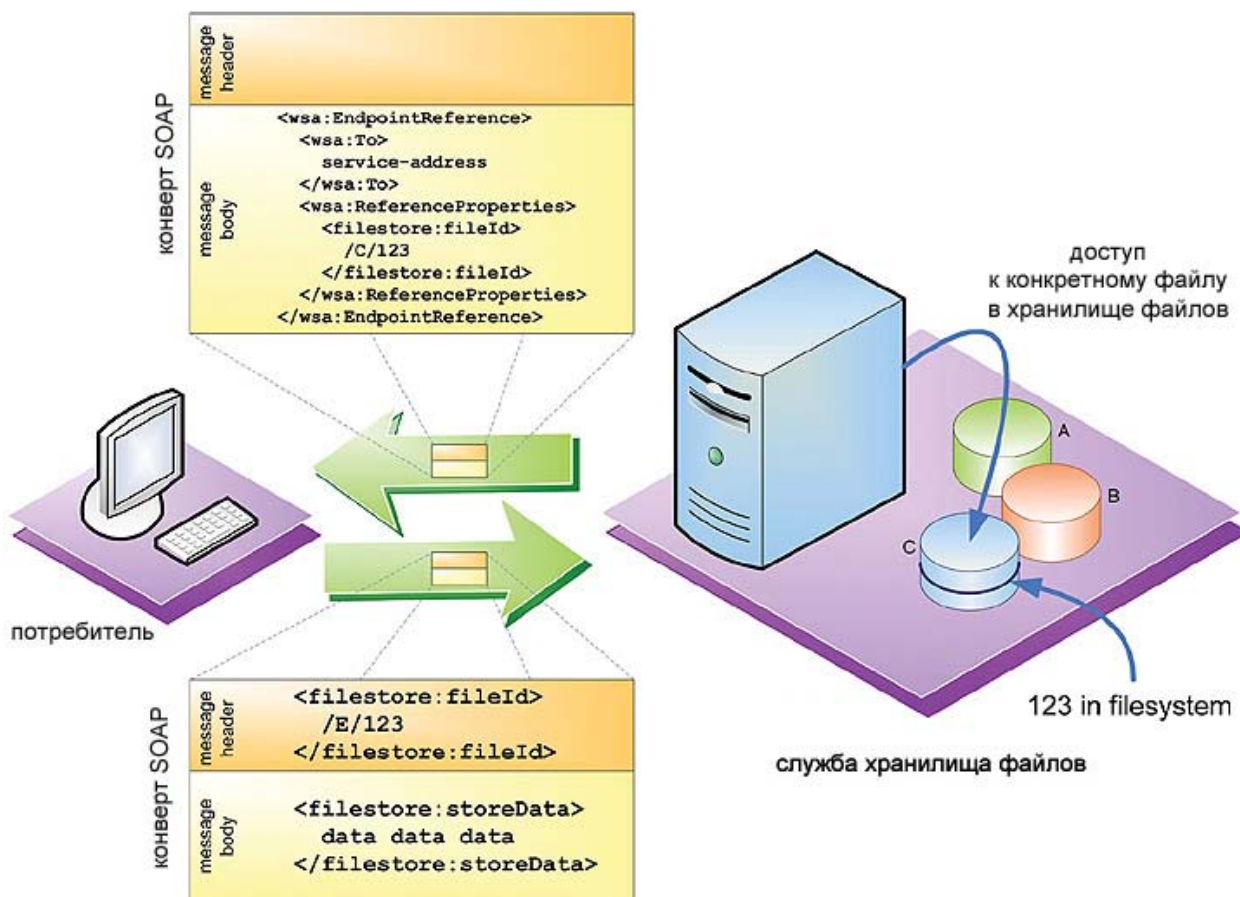


Рис. 3

Хотя с точки зрения протокола SOAP реализация схемы WSRF выглядит вполне дружелюбной (используются SOAP-заголовки и WS-Addressing), разработчики, тем не менее, должны проявлять осторожность и не разрушить инкапсуляцию из-за прямого экспонирования частных ресурсов предприятия в более широкой сети. Основная опасность, связанная с подходом WSRF, состоит в том, что он как раз потворствует именно такой манере работы. Это в свою очередь ведёт к созданию хрупких и трудных для сопровождения приложений. Ключевая слабость такого подхода проявляется в том, что при необходимости как-то развить возможности службы (например, когда реализация хранилища файлов системы мигрирует от одной файловой системы к системе управления базой данных) идентичность информации, содержащейся в метаданных ссылки на крайнюю точку, может оказаться устаревшей. Как следствие этого, сессия с сохранением состояния не достигнет успеха. Вот почему дополнительные механизмы, такие, как управление сроком жизни и возобновление ссылок являются необходимыми частями платформы WSRF. Хотя и возможно исключить такие проблемы, используя логические идентификаторы (которые разрешаются службой в физические ресурсы), это не предписывается спецификациями WSRF. (Заметим, что подход WS-Context не страдает таким недостатком, поскольку контексты являются сущностями третьей стороны, всецело отделёнными от реализации любой службы).

В рассматриваемом нами ограниченном сценарии, подход WS-Context полностью не отличим от подхода WSRF. Контекст генерируется некоторыми вспомогательными средствами (например, службой контекста) и встраивается в блок SOAP-заголовка с

каждым сообщением уровня приложения, посылаемым хранилищу файлов. Последнее выполняет действие, соответствующее предписанию в сообщении, обрабатывая при этом информацию о контексте, чтобы обеспечить корректное состояние и доступ к ресурсам, используемым для обслуживания действия.

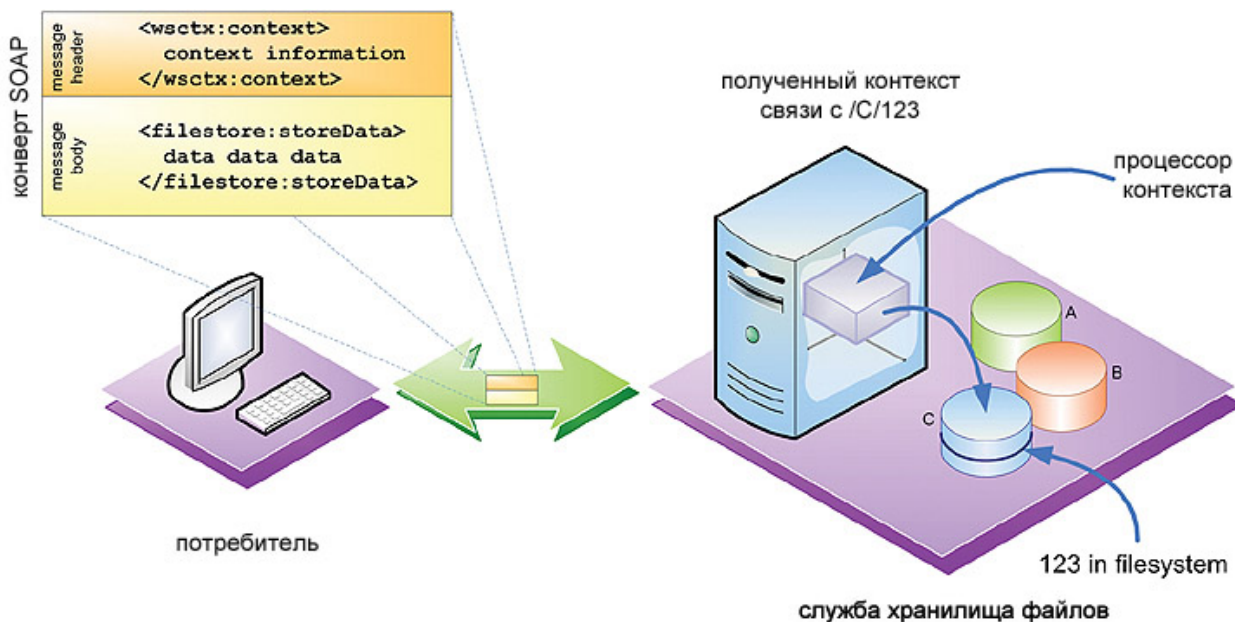


Рис. 4

В отличие от принятого в подходе WSRF способа идентификации, контекст, обозначенный на рис. 4, не является идентификатором какого-либо внутреннего ресурса, а это - внешняя сущность, которая даёт возможность логически связывать действия. Подход WS-Context не пытается моделировать ресурсы серверной части, поскольку они не рассматриваются в качестве его объектов, и, тем не менее, взаимодействия, сохраняющие состояние, всё же могут быть поддержаны. Так как в основе WS-Context-подхода лежит посыл, что реализация службы является конфиденциальной, то выбор способа использования контекстной информации для корреляции сообщений в ходе взаимодействий, сохраняющих состояние, остаётся за архитектурой службы. Это означает, что в то время, как WS-Context-осведомляемые службы являются интероперабельными, от разработчиков не требуется выбирать реализацию службы. Так, например, подход WS-Context предполагает, что Web-служба может независимо эволюционировать, и при этом никакая информация (логическая или иная) о конфигурации служебных переходов не выходит за её границы. Поскольку контекстная информация существует независимо от любой контекстно-уведомляемой службы, такие службы могут развиваться так, как это целесообразно, не подвергая опасности правильность будущих контекстуализированных взаимодействий.

### Масштабирование WSRF и WS-Context

Если воспользоваться подходом WSRF разумно, то можно поддерживать парные взаимодействия способом, подобным подходу WS-Context, при этом содержимое элемента `<ReferenceProperties/>` описания WSRF-конечной точки будет использоваться вместо WS-Context-контекста. При определённых стеснённых обстоятельствах WSRF-подход

является даже более простым решением, поскольку он не требует дополнительных протокольных актов, в то время как большинство ориентированных на WS-Context служб спроектированы так, чтобы принимать участие в распределённых активностях, и поэтому нуждаются в службе управления контекстом, или внутренних средствах генерации контекста.

Однако, в отличие от подхода WS-Context, который трактует контекст, как некую внешне разделяемую сущность, WSRF-модель плохо подходит для масштабирования уже состоявшихся простых клиент-серверных взаимодействий, поскольку ресурсы идентифицируются посредством информации конкретной службы, которая используется для контекстуализации взаимодействий. Для иллюстрации этого принципа обратимся к рис. 5, 6.

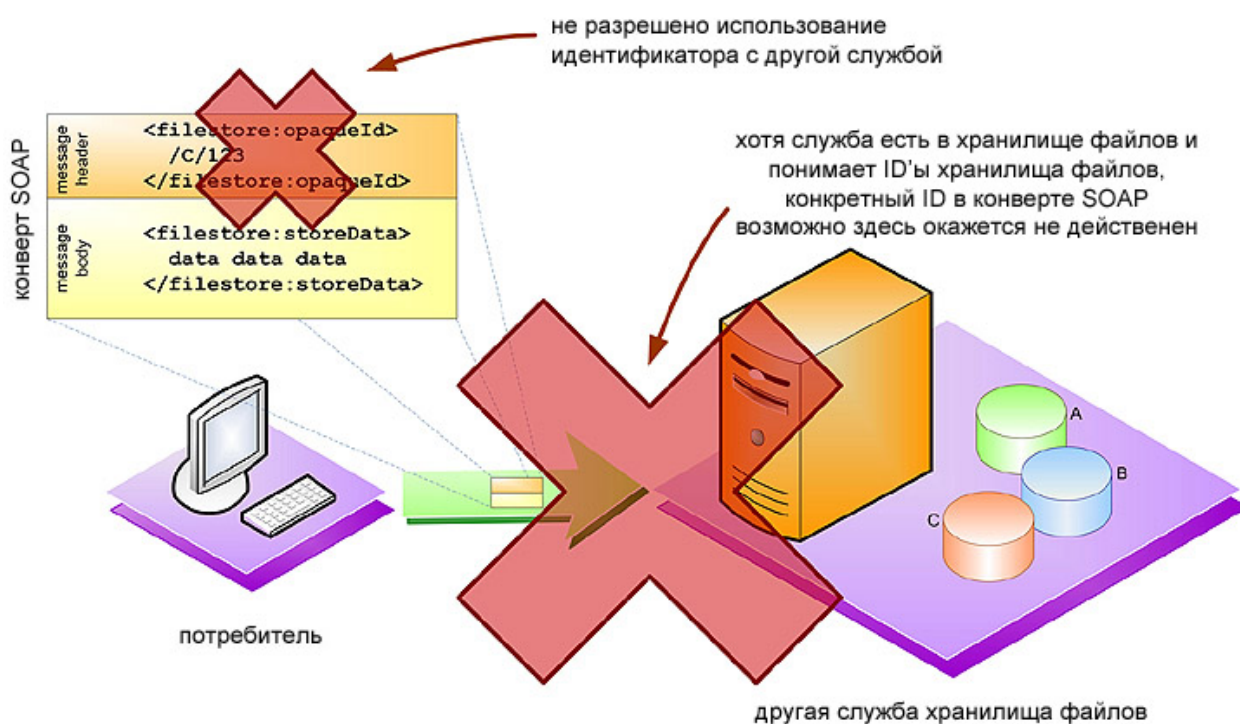


Рис. 5

Рассматривая рис. 5, мы убеждены, что подобный указателю механизм, который фундаментален в концепции подхода WSRF, действителен, когда используется для связи с ресурсом, обслуживаемым какой-то конкретной службой. Однако, поскольку эта ссылка на крайнюю точку привязана к конкретной службе, те же самые ресурсно-связанные метаданные не могут использоваться при коммуникации с другой службой.

Кроме того, поскольку нет никаких средств, чтобы предотвратить хранение структур WS-Addressing с метаданными, касающимися ресурсов и информации о крайней точке службы, могут сформироваться долгоживущие взаимозависимости между ресурсами. Такие взаимозависимости трудно обслуживать в крупномасштабных системах, и это оказывается причиной хрупкости приложений.

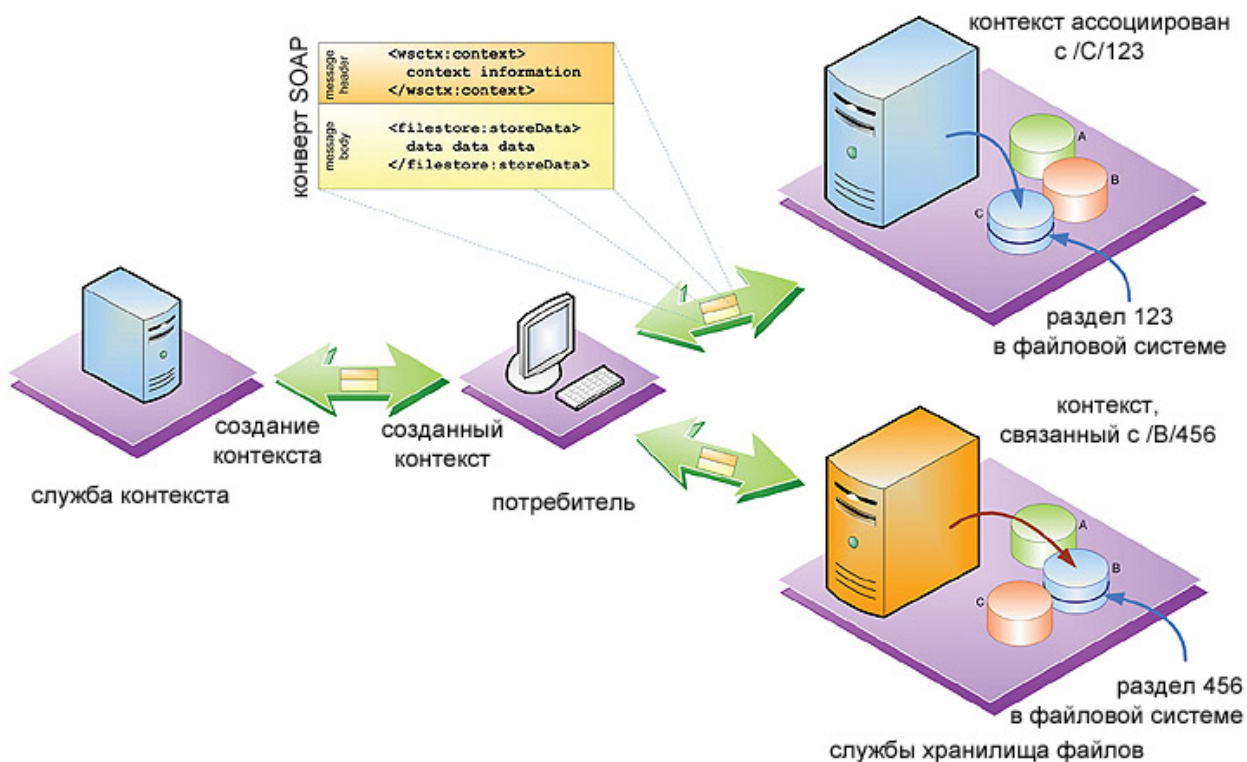


Рис. 6

На рис. 6 показано, что поскольку контекст моделируется как известная, стандартизированная, внешняя сущность, то любая WS-Context-осведомляемая служба, которая получает его, сможет применить контекстную информацию для своих собственных внутренних процессов. При условии, что данный контекст явно управляем и является внешним для любой отдельной службы, он видим и действенен для всех служб внутри данной активности. После получения контекста служба может использовать содержащуюся в нём информацию для корреляции сообщений, направляемых внутренним ресурсам, а также воспользоваться ею для обнаружения более широкого контекста приложения, внутри которого данная активность будет выполнена (то есть, там, где участвуют другие службы) и таким образом возможны распределённые активности с сохранением состояния.

Хотя распределённые активности и можно реализовывать, следуя модели подхода WSRF (путём ручного распространения всех ссылок на крайние точки при обращениях ко всем используемым службам), всё же следует признать следующее: установление области действия распределённого приложения на основе использования наборов двухточечных адресных сочетаний по своему существу весьма затруднительно. Это очень быстро ведет к комбинаторному взрыву ссылок на крайние точки, которые нужно обрабатывать, распространять и сохранять актуальными. Напротив, только единственная сущность, а именно, контекст, требуется при подходе WS-Context.

### Заключение

Хотя как подход WS-Context, так и подход WSRF могут использоваться для поддержки взаимодействий, сохраняющих состояние между Web-службами и их

потребителями, модели, которые они используют, очень разнятся, и, как следствие этого, сценарии, в которых они лучше всего применимы, также различаются.

Подход WSRF базируется на схеме адресации внутренних ресурсов, ассоциируемых с Web-службами. Эта адресная информация может быть использована как средство корреляции и маршрутизации обменов сообщениями с этими внутренними ресурсами и, таким образом, как средство обеспечения связи с сохраняемым состоянием.

Подход WS-Context предполагает, что внутренние детали реализации службы являются конфиденциальными. Она не доступна для какого-либо разрушительного анализа и имеет дело только с управлением контекстом и передачей контекстов службам. Что точно делается для отображения требования конкретного сообщения уровня приложения, дополненного контекстом, на конкретных внутренних ресурсах, скрыто от окружения.

Для одиночных взаимодействий потребитель-сервер, подход WSRF определённо легче. Тем не менее, при взаимодействиях, вовлекающих множество служб, подход WS-Context- легко масштабируется для поддержки распределённых активностей. Поэтому подход WSRF, как решение типа "точка-точка", возможно более удобен при интеграции двух систем, но в общем случае, с системами, составленными из многих Web-служб, подход WS-Context представляется естественным выбором.

### **Благодарности**

Авторы благодарят профессора Пауля Ватсона (Paul Watson, University of Newcastle upon Tyne) за рекомендации, высказанные в связи с данной статьёй.

### **Ссылки**

- *OASIS(WS-CAF), Web services Context (WS-CTX):*  
<http://www.iona.com>
- *Web services Resource Framework (WSRF). 2004:* [www.globus.org/wsrf](http://www.globus.org/wsrf)
- Parastatidis, S., et al. (2003). "A Grid Application Framework Based on Web Services Specifications and Practices. [www.neresc.ac.uk/ws-gaf](http://www.neresc.ac.uk/ws-gaf)
- Tuecke, S., et al. (2003). "Open Grid Services Infrastructure (OGSI) - Version 1.0".  
<https://forge.gridforum.org/projects/ogsi-wg>
- Frey, J., et al. (2004) "Modeling Stateful Resources with Web Services."
- *Web Services Addressing (WS-Addressing):*  
<http://msdn.microsoft.com/ws/2003/03/ws-addressing>.