WildBoss Pro

# Руководство Администратора

# Содержание

# 1 Назначение документа

Настоящий документ представляет собой руководство по установке, администрированию и конфигурированию WildBoss Pro.

## 1.1 Предварительные требования

Прежде чем продолжить, необходимо знать, как скачать, установить и запустить WildBoss Pro. Для получения подробной информации по процедуре скачивания, установки и запуска WildBoss Pro необходимо обратиться к документу «Руководство по началу работы».

## 1.2 Примеры в настоящем руководстве

Примеры в настоящем руководстве в основном представлены в виде выдержек из файла конфигурации XML или файла с использованием представления не типизированной модели управления.

# 2 Основные концепции управления

## 2.1 Режим работы

WildBoss Pro может быть загружен в двух различных режимах. Управляемый домен позволяет запускать многосерверную топологию и управлять ею. В качестве альтернативы допускается запустить автономный экземпляр сервера.

### 2.1.1 Автономный сервер

Во многих случаях возможности централизованного управления, доступные через управляемый домен, не требуются. В этих случаях экземпляр WildBoss Pro может быть запущен как автономный сервер («standalone server»). Автономный экземпляр сервера - это независимый процесс. Автономные экземпляры могут быть запущены с помощью сценариев запуска «standalone.sh» или «standalone.bat». Если запущено более одного автономного экземпляра и требуется управление несколькими серверами, пользователь несет ответственность за координацию управления между серверами. Например, для развертывания приложения на всех автономных серверах пользователю потребуется индивидуально развернуть приложение на каждом сервере. Вполне возможно запустить несколько автономных экземпляров сервера и объединить их в кластер высокой производительности.

### 2.1.2 Управляемый домен

Одной из основных новых функций WildBoss Pro является возможность управлять несколькими экземплярами WildBoss Pro из одной точки управления. Совокупность таких серверов называется членами «домена», в котором один процесс контроллера домена выступает в качестве центральной точки управления. Все экземпляры WildBoss Pro в домене используют общую политику управления, и контроллер домена следит за тем, чтобы каждый сервер был настроен в соответствии с этой политикой. Домены могут охватывать несколько физических (или виртуальных) машин, при этом все экземпляры WildBoss Pro на данном хосте находятся под управлением специального процесса контроллера хоста. Один экземпляр контроллера хоста настроен на выполнение функций центрального Контроллера домена. Главный контроллер на каждом хосте взаимодействует с контроллером домена для управления жизненным циклом экземпляров сервера приложений, запущенных на этом хосте, и для оказания помощи домену в управлении ими. Контроллер домена управляет сервером приложений, запущенным на этом хосте.

Когда запускается управляемый WildBoss Pro домен на хосте (с помощью сценариев запуска «domain.sh» или «domain.bat»), планируется запуск контроллера хоста и, как правило, по крайней мере один экземпляр WildBoss Pro. На одном из хостов контроллер хоста должен быть настроен для выполнения функций контроллера домена. Подробности смотрите в разделе «Настройка домена».

Далее по тексту приведен пример управляемой топологии домена:

```
[images/DC-HC-Server.png]
```

#### 2.1.2.1 Хост

Каждый блок «Хост» на приведенной выше диаграмме представляет собой физический или виртуальный хост. Физический хост может содержать ноль, один или несколько экземпляров сервера.

#### 2.1.2.2 Хост контроллера

Когда скрипт «domain.sh» или «domain.bat» запускается на хосте, запускается процесс, известный как контроллер хоста. Контроллер хоста отвечает исключительно за управление сервером, он сам не обрабатывает рабочие нагрузки сервера приложений. Хост-контроллер отвечает за запуск и остановку отдельных процессов сервера приложений, которые выполняются на его хосте, и взаимодействует с доменом «Контроллер», чтобы помочь управлять ими.

Каждый хост-контроллер по умолчанию считывает свою конфигурацию из domain/configuration/host.xml, файл находится в распакованной установке WildBoss Pro в файловой системе ее хоста. Тот самый «host.xml» файл содержит информацию о конфигурации, относящуюся к конкретному хосту. Преимущественно:

– список имен реальных экземпляров WildBoss Pro, которые предназначены для запуска в этой установке;

– конфигурация того, как хост-контроллер должен связаться с контроллером домена, чтобы зарегистрироваться и получить доступ к конфигурации домена. Это может быть либо конфигурация того, как найти удаленный контроллер домена и связаться с ним, либо конфигурация, позволяющая хост-контроллеру самому выступать в качестве контроллера домена;

– настройка элементов, специфичных для локальной физической установки. Например, именованные определения интерфейсов, объявленные в «domain.xml» (см. ниже) может быть сопоставлен с реальной спецификой машины. IP-адрес в «host.xml». Абстрактные имена путей в «domain.xml» могут быть сопоставлены с реальными путями к файловой системе в «host.xml».

#### 2.1.2.3 Контроллер домена

Один экземпляр хост-контроллера настроен таким образом, чтобы выступать в качестве центральной точки управления для всего домена, т.е. быть контроллером домена. Основной обязанностью контроллера домена является поддержание политики централизованного управления доменом, обеспечение того, чтобы все контроллеры хоста были осведомлены о ее текущем содержании, и оказание помощи контроллерам хоста в обеспечении того, чтобы все запущенные экземпляры сервера приложений были настроены в соответствии с этой политикой. Эта политика централизованного управления по умолчанию хранится в файле domain/configuration/domain.xml в распакованной установке WildBoss Pro на файловой системе хоста контроллера домена.

Файл «domain.xml» должен находиться в каталоге «domain/configuration» установки, предназначенной для запуска контроллера домена. Он не обязательно должен присутствовать в установках, которые не предназначены для запуска контроллера домена, т.е. в тех, чей хост-контроллер настроен для связи с удаленным контроллером домена. Наличие файла «domain.xml» на таком сервере не наносит вреда.

Тот самый «domain.xml» файл включает в себя, среди прочего, конфигурацию различных «profiles», которые Экземпляры WildBoss Pro в домене можно настроить для запуска. Конфигурация профиля включает в себя подробную конфигурацию различных подсистем, составляющих этот профиль (например, встроенного JBossWeb instance – это подсистема; диспетчер транзакций JBoss TS – это подсистема и т.д.). Конфигурация домена также включает определение групп сокетов, которые могут быть открыты этими подсистемами. Конфигурация домена также включает определение «группы серверов».

##### 2.1.2.3.1 Группа серверов

«Группа серверов» - это набор экземпляров сервера, которые будут управляться и настраиваться как один. В управляемом домене каждый экземпляр сервера приложений

является членом группы серверов. (Даже если в группе есть только один сервер, сервер все равно является членом группы.) Ответственность за то, чтобы все серверы в группе серверов имели согласованную конфигурацию, лежит на домене Контроллер и хост-контроллеры должны обеспечивать согласованную конфигурацию всех серверов. Все они должны быть настроены с использованием одного и того же профиля и иметь одинаковое содержимое для развертывания.

В домене может быть несколько групп серверов. На приведенной выше диаграмме показаны две группы серверов: «ServerGroupA» и «ServerGroupB». Различные группы серверов могут быть сконфигурированы с различными профилями и развертываниями; например, в домене с различными уровнями серверов, предоставляющих различные услуги. Различные группы серверов также могут использовать один и тот же профиль и иметь одинаковые возможности развертывания; например, для поддержки сценариев последовательного обновления приложений, в которых можно избежать полного отключения обслуживания, сначала обновив приложение на одной группе серверов, а затем обновив вторую группу серверов.

Пример определения группы серверов выглядит следующим образом:

```
<server-group name="main-server-group" profile="default">
 <socket-binding-group ref="standard-sockets"/>
   <deployments>
     <deployment name="foo.war_v1" runtime-name="foo.war" />
     <deployment name="bar.ear" runtime-name="bar.ear" />
   </deployments>
 </server-group>
```

Конфигурация группы серверов включает в себя следующие обязательные атрибуты:

– имя — название группы серверов;

– профиль — имя профиля, которое должно быть запущено на серверах в группе.

Кроме того, доступны следующие дополнительные элементы:

– socket-binding-group — указывает имя группы привязки сокетов по умолчанию, используемой на серверах в группе. Может быть переопределено для каждого сервера в разделе «host.xml». Если оно не указано в элементе «server-group» , оно должно быть указано для каждого сервера в разделе «host.xml»;

– deployments — содержимое развертывания, которое должно быть развернуто на серверах в группе;

– deployment-overlays — наложения и связанные с ними развертывания;

– system-properties — системные свойства, которые должны быть установлены на всех серверах в группе;

– jvm — настройки «jvm» по умолчанию для всех серверов в группе. Хост-контроллер объединит эти настройки с любыми, указанными в «host.xml», чтобы получить настройки, которые будут использоваться для запуска JVM сервера. Дополнительные сведения см. в разделе «Настройки JVM».

#### 2.1.2.4 Сервер

Каждый «Сервер» на приведенной выше диаграмме представляет собой реальный экземпляр сервера приложений. Сервер запускается в процессе JVM, отдельном от контроллера хоста. Контроллер хоста отвечает за запуск этого процесса (в управляемом домене конечный пользователь не может напрямую запустить серверный процесс из командной строки). Контроллер хоста синтезирует конфигурацию сервера, комбинируя элементы конфигурации всего домена (из «domain.xml») и конфигурации конкретного хоста (из «host.xml»).

### 2.1.3 Выбор между запуском автономных серверов или управляемого домена

Какие варианты использования подходят для управляемого домена, а какие - для автономных серверов? Управляемый домен – это прежде всего скоординированное управление несколькими серверами. С его помощью WildBoss Pro предоставляет центральную точку, с помощью которой пользователи могут управлять несколькими серверами, с широкими возможностями для обеспечения согласованности конфигураций этих серверов и возможностью внесения изменений в конфигурацию (включая развертывание) на серверах скоординированным образом.

Важно понимать, что выбор между управляемым доменом и автономными серверами зависит от того, как управляются ваши серверы, а не от того, какими возможностями они обладают для обслуживания запросов конечных пользователей. Это различие особенно важно, когда речь идет о кластерах высокой доступности. Важно понимать, что функциональность HA не зависит от работы автономных серверов или управляемого домена. То есть группа автономных серверов может быть сконфигурирована для формирования кластера HA. Доменный и автономный режимы определяют способ управления серверами, а не то, какие возможности они предоставляют.

Учитываю все это:

– установка одного сервера ничего не дает по сравнению с запуском в управляемом домене, поэтому лучше использовать автономный сервер;

– в производственных средах с несколькими серверами выбор между использованием управляемого домена и автономных серверов зависит от того, хочет ли пользователь использовать возможности централизованного управления, предоставляемые управляемым доменом. Некоторые предприятия разработали свои собственные сложные многосерверные средства управления и могут с удобством координировать изменения в нескольких независимых экземплярах WildBoss Pro. Для таких предприятий многосерверная архитектура, состоящая из отдельных автономных серверов, является хорошим вариантом;

– запуск автономного сервера лучше подходит для большинства сценариев разработки. Любая индивидуальная конфигурация сервера, которая может быть реализована в управляемом домене, также может быть реализована на автономном сервере, поэтому, даже если разрабатываемое приложение в конечном итоге будет запущено в производство в управляемом домене, большая часть (вероятно, большая часть) разработки может быть выполнена с использованием автономного сервера;

– запуск режима управляемого домена может быть полезен в некоторых сложных сценариях разработки, например, в тех, которые предполагают взаимодействие между несколькими экземплярами WildBoss Pro. Разработчики могут обнаружить, что настройка различных серверов в качестве членов домена является эффективным способом запуска многосерверного кластера.

## 2.2 Общие концепции конфигурации

Как для управляемого домена, так и для автономного сервера применяется ряд общих принципов настройки.

### 2.2.1 Расширения

Расширение - это модуль, который расширяет основные возможности сервера. Ядро WildBoss Pro очень простое и легкое; большинство возможностей, которые связывают с сервером приложений, предоставляются с помощью расширений. Расширение упаковано в виде модуля в папке «modules». Пользователь указывает на то, что определенные расширения

должны быть доступны, в том числе <расширения/> элемент называя ее модуль в domain.xml или standalone.xml файл.

```
<extensions>
  [...]
   <extension module="org.jboss.as.transactions"/>
   <extension module="org.jboss.as.webservices" />
   <extension module="org.jboss.as.weld" />
  [...]
<extension module="org.WildBoss Pro.extension.undertow"/>
 </extensions>
```

### 2.2.2 Профили и подсистемы

Наиболее важной частью конфигурации в «domain.xml» и «standalone.xml» является настройка одного (в «standalone.xml») или нескольких (в «domain.xml») «профилей». Профиль - это именованный набор конфигураций подсистемы. Подсистема - это дополнительный набор возможностей, добавляемый к основному серверу с помощью расширения (см. пункт «Расширения»). Подсистема предоставляет возможности обработки сервлетов; подсистема предоставляет контейнер Jakarta Enterprise Beans; подсистема предоставляет транзакции Jakarta и т.д. Профиль - это именованный список подсистем с подробной информацией о конфигурации каждой из них. Профиль с большим количеством подсистем позволяет создать сервер с большим набором возможностей. Профиль с небольшим, сфокусированным набором подсистем будет обладать меньшими возможностями, но и занимать меньшую площадь.

Содержимое индивидуальной конфигурации профиля выглядит в основном одинаково в «domain.xml» и «standalone.xml». Единственное различие заключается в том, что «standalone.xml» разрешено иметь только один элемент профиля (профиль, который будет запускаться сервером), в то время как «domain.xml» может быть много профилей, каждый из которых может быть сопоставлен одной или нескольким группам серверов.

Содержимое конфигураций отдельных подсистем выглядит точно так же между «domain.xml» и «standalone.xml».

### 2.2.3 Пути

Логическое имя пути к файловой системе. Все «domain.xml», «host.xml» и «standalone.xml» конфигурации содержат раздел, в котором могут быть объявлены пути. Затем другие разделы конфигурации могут ссылаться на эти пути по их логическому имени, вместо того чтобы включать полную информацию о пути (которая может отличаться на разных компьютерах). Например, конфигурация подсистемы ведения журнала содержит ссылку на путь «jboss.server.log.dir», который указывает который указывает на каталог «log» сервера.

```
<file relative-to="jboss.server.log.dir" path="server.log"/>
```

WildBoss Pro автоматически предоставляет ряд стандартных путей без какой-либо необходимости для пользователя настраивать их в файле конфигурации:

– `jboss.home.dir` - корневой каталог дистрибутива WildBoss Pro;

– `user.home` - домашний каталог пользователя;

– `user.dir` - текущий рабочий каталог пользователя;

– `java.home` - каталог установки java;

– `jboss.server.base.dir` - корневой каталог для отдельного экземпляра сервера;

– jboss.server.config.dir - каталог, который сервер будет использовать для хранения конфигурационных файлов;

– jboss.server.data.dir - каталог, который сервер будет использовать для постоянного хранения файлов данных;

– jboss.server.log.dir - каталог, который сервер будет использовать для хранения файлов журнала;

– jboss.server.temp.dir - каталог, который сервер будет использовать для временного хранения файлов;

– jboss.controller.temp.dir - каталог, который сервер будет использовать для временного хранения файлов;

– jboss.domain.servers.dir - каталог, в котором хост-контроллер создаст рабочую область для отдельных экземпляров сервера (только в режиме управляемого домена).

Пользователи могут добавлять свои собственные пути или переопределять все, кроме первых 5 (пяти) из вышеперечисленных, добавив элемент <path/> в свой файл конфигурации.

```
<path name="example" path="example" relative-to="jboss.server.data.dir"/>
```

Этими атрибутами являются:

– name - имя пути;

– path - фактический путь к файловой системе. Рассматривается как абсолютный путь, если не указан атрибут 'relative-to', и в этом случае значение рассматривается как относительное к этому пути;

– relative-to - (необязательно) имя другого ранее указанного пути или одного из стандартных путей, предоставляемых системой.

Элемент <path/> в a domain.xml не должен содержать ничего, кроме атрибута name; т.е. он не должен содержать никакой информации, указывающей на фактический путь к файловой системе

Элемент <path/> в «domain.xml» не должен содержать ничего, кроме атрибута «name», т.е. он не должен содержать никакой информации, указывающей, каков фактический путь к файловой системе.

```
<path name="x"/>
```

Такая конфигурация просто сообщает: «Существует путь с именем «x», на который могут ссылаться другие части конфигурации «domain.xml». Фактическое местоположение файловой системы, на которое указывает «x», зависит от хоста и будет указано в файле «host.xml» каждой машины». Если используется этот подход, в файловой системе каждого компьютера должен быть элемент «path host.xml», который определяет фактический путь к файловой системе.

```
<path name="x" path="/var/x" />
```

Элемент <path/> в «standalone.xml» должен содержать указание фактического пути к файловой системе.

### 2.2.4 Интерфейсы

Логическое имя для сетевого интерфейса/IP-адреса/имени хоста, к которому могут быть привязаны сокеты. Все «domain.xml», «host.xml» и «standalone.xml» конфигурации содержат раздел, в котором могут быть объявлены интерфейсы. Затем другие разделы конфигурации могут ссылаться на эти интерфейсы по их логическому имени, вместо того чтобы включать полную информацию об интерфейсе (которая может отличаться на разных компьютерах). Конфигурация интерфейса включает в себя логическое имя интерфейса, а также информацию, определяющую критерии, которые следует использовать для определения фактического физического адреса для использования. Дополнительные сведения см. в разделе «Интерфейсы и порты».

Элемент <interface/> в «domain.xml» не должен содержать ничего, кроме атрибута «name», т.е. он не должен содержать никакой информации, указывающей, каков фактический IP-адрес, связанный с именем.

```
<interface name="internal"/>
```

Такая конфигурация просто сообщает: «Существует интерфейс с именем «internal», на который могут ссылаться другие части конфигурации «domain.xml». Фактический IP-адрес, на который указывает «internal», зависит от хоста и будет указан в файле «host.xml» каждой машины». Если этот подход используется, должно такой элемент интерфейса «host.xml» каждая машина, которая определяет критерии для определения IP адрес.

```
<interface name="internal">
  <nic name="eth1"/>
</interface>
```

Элемент <интерфейс/> в «standalone.xml» должен содержать критерии для определения IP -адреса.

Более подробную информацию см. в объявлениях интерфейса.

### 2.2.5 Привязки сокетов и группы привязок сокетов

Привязка сокета - это именованная конфигурация для сокета.

Конфигурации domain.xml и standalone.xml содержат раздел, в котором могут быть объявлены именованные конфигурации сокетов. Другие разделы конфигурации могут ссылаться на эти сокеты по их логическому имени вместо того, чтобы включать полную информацию о конфигурации сокета (которая может отличаться на разных компьютерах). Более подробную информацию см. в разделе «Группы привязки сокетов».

### 2.2.6 Системные свойства

Значения системных свойств могут быть заданы в нескольких местах в «domain.xml», «host.xml» и «standalone.xml». Значения в «standalone.xml» задаются в процессе загрузки сервера. Значения в «domain.xml» и «host.xml» применяются к серверам при их запуске.

Когда системное свойство настроено в «domain.xml» или «host.xml», серверы, к которым оно в конечном итоге применяется, зависят от того, где оно установлено. Установка системного свойства в дочернем элементе непосредственно в корневой папке «domain.xml» приводит к тому, что свойство устанавливается на всех серверах. Установка этого параметра в элементе <system-property/> внутри элемента <server-group/> в «domain.xml» приводит к тому, что свойство устанавливается на всех серверах в группе. Установка этого параметра в дочернем элементе непосредственно под корневым «host.xml» приводит к тому, что свойство

устанавливается на всех серверах, управляемых контроллером хоста этого хоста. Наконец, установка этого параметра в элементе <system-property/> внутри элемента <server/> в «host.xml» приводит к тому, что свойство устанавливается на этом сервере. Одно и то же свойство может быть настроено в нескольких местах, при этом значение в элементе <server/> будет иметь приоритет над значением, указанным непосредственно в корневом элементе «host.xml», значение в элементе «host.xml» будет иметь приоритет над всем, что указано в элементе «domain.xml», и значением в элементе <server-group/>.

### 2.2.7 Файлы конфигурации скрипта

Скрипты расположены в каталоге «$JBOSS_HOME/bin». В этом каталоге расположены файлы конфигурации скриптов для автономного запуска и запуска домена для каждой платформы. Эти файлы можно использовать для настройки вашей среды без необходимости редактирования самих скриптов. Например, вы можете настроить переменную окружения «JAVA_OPTS» для настройки JVM перед запуском контейнера.

Файлы конфигурации автономного скрипта:

– «standalone.conf» вызывается из «standalone.sh»;

– «standalone.conf.bat» вызывается из «standalone.bat»;

– «standalone.conf.ps1» вызывается из «standalone.ps1».

Файлы конфигурации доменного скрипта:

– «domain.conf» вызывается из «domain.sh»;

– «domain.conf.bat» вызывается из «domain.bat»;

– «domain.conf.ps1» вызывается из «domain.ps1».

По умолчанию они находятся в каталоге «$JBOSS_HOME/bin». Однако вы можете задать переменную среды «STANDALONE_CONF» для автономных серверов или переменную среды «DOMAIN_CONF» для доменных серверов со значением абсолютного пути к файлу.

#### 2.2.7.1 Общие файлы конфигурации скрипта

Общие файлы конфигурации вызываются из каждого скрипта в каталоге «$JBOSS_HOME/bin». Хотя эти файлы конфигурации отсутствуют в каталоге по умолчанию, их можно добавить. Вы можете просто добавить конфигурационный файл «common.conf» для типа скрипта, который необходимо выполнить, и все скрипты в каталоге будут вызывать конфигурационный скрипт

– «common.conf» для bash-скриптов;

– «common.conf.bat» для пакетных сценариев Windows;

– «common.conf.ps1» для скриптов PowerShell.

Также имеется возможность задать переменную среды «COMMON_CONF», чтобы этот конфигурационный скрипт работал за пределами каталога «$JBOSS_HOME/bin».

**Внимание:** если предоставляется общий файл конфигурации, он будет вызван перед файлами конфигурации «standalone» и доменного скрипта. Например, при вызове «standalone.sh» вызывает «common.conf», а затем «standalone.conf».

## 2.3 Ресурсы управления

Когда WildBoss Pro анализирует пользовательские файлы конфигурации при загрузке или когда пользователь использует один из клиентов управления AS, пользователь добавляет, удаляет или изменяет ресурсы управления во внутренней модели управления AS. Ресурс управления WildBoss Pro обладает характеристиками, приведенными далее по тексту.

### 2.3.1 Адрес

Все ресурсы управления WildBoss Pro организованы в виде дерева. Путь к узлу в дереве для конкретного ресурса - это его адрес. Каждый сегмент в адресе ресурса представляет собой пару ключ/значение:

– ключом является тип ресурса в контексте его родительского элемента. Так, например, корневой ресурс для автономного сервера имеет дочерние элементы типа «subsystem», «interface», «socket-binding» и т.д. Ресурс для подсистемы, предоставляющий возможности веб-сервера AS, имеет дочерние элементы типа «connector» и «virtual-server». Ресурс для подсистемы, предоставляющий возможности сервера обмена сообщениями AS, имеет, среди прочего, дочерние элементы типа «jms-queue» и «jms-topic»;

– значение - это имя конкретного ресурса данного типа, например «web» или «messaging» для подсистем или «http» или «https» для соединителей веб-подсистем.

Полный адрес ресурса - это упорядоченный список пар «ключ/значение», которые ведут от корня дерева к ресурсу. Обычно элементы адреса разделяются символом «/», а ключ и значение - символом «=»:

– «`/subsystem=undertow/server=default-server/http-listener=default`»;

– «`/subsystem=messaging/jms-queue=testQueue`»;

– «`/interface=public`».

При использовании HTTP API для разделения ключа и значения используется символ «/» вместо символа «=»:

– «`http://localhost:9990/management/subsystem/undertow/server/default-server/http-listener/default`»;

– «`http://localhost:9990/management/subsystem/messaging/jms-queue/testQueue`»;

– «`http://localhost:9990/management/interface/public`».

### 2.3.2 Операции

Запрос или изменение состояния ресурса выполняется с помощью операции. Операция имеет следующие характеристики:

– строковое имя;

– ноль или более именованных параметров. Каждый параметр имеет строковое имя и значение типа «org.jboss.dmr.ModelNode» (или, при вызове через интерфейс командной строки, текстовое представление ModelNode; при вызове через HTTP API используется JSON-представление ModelNode.) Параметры могут быть необязательными;

– возвращаемое значение, которое будет иметь тип «org.jboss.dmr.ModelNode» (или, при вызове через интерфейс командной строки, текстовое представление ModelNode; при вызове через HTTP API, JSON-представление ModelNode).

Каждый ресурс, за исключением корневого, будет иметь операцию добавления и должен иметь операцию удаления. Параметры операции добавления варьируются в зависимости от ресурса. Операция удаления не имеет параметров.

Существует также ряд «глобальных» операций, которые применяются ко всем ресурсам. Более подробную информацию см. в разделе «Глобальные операции».

Сами операции, поддерживаемые ресурсом, могут быть определены путем вызова операции: операции чтения имен операций. Как только имя операции известно, подробные сведения о ее параметрах и возвращаемом значении могут быть определены путем вызова операции чтения описания операций. Например, чтобы узнать названия операций, выполняемых корневым ресурсом для автономного сервера, а затем узнать полную информацию об одной из них с помощью интерфейса командной строки, необходимо:

```
[standalone@localhost:9990 /] :read-operation-names
{
  "outcome" => "success",
  "result" => [
    "add-namespace",
    "add-schema-location",
    "delete-snapshot",
    "full-replace-deployment",
    "list-snapshots",
    "read-attribute",
    "read-children-names",
    "read-children-resources",
    "read-children-types",
    "read-config-as-xml",
    "read-operation-description",
    "read-operation-names",
    "read-resource",
    "read-resource-description",
    "reload",
    "remove-namespace",
    "remove-schema-location",
    "replace-deployment",
    "shutdown",
    "take-snapshot",
    "upload-deployment-bytes",
    "upload-deployment-stream",
    "upload-deployment-url",
    "validate-address",
    "write-attribute"
  ]
}
[standalone@localhost:9990 /] :read-operation-description(name=upload-
deployment-url)
{
  "outcome" => "success",
  "result" => {
    "operation-name" => "upload-deployment-url",
    "description" => "Indicates that the deployment content available at the
    included URL should be added to the deployment content repository. Note that
    this operation does not indicate the content should be deployed into the
    runtime.",
    "request-properties" => {"url" => {
      "type" => STRING,
      "description" => "The URL at which the deployment content is available for
      upload to the domain's or standalone server's deployment content
      repository.. Note
      that the URL must be accessible from the target of the operation (i.e. the
      Domain
      Controller or standalone server).",
      "required" => true,
      "min-length" => 1,
      "nillable" => false
    }},
    "reply-properties" => {
      "type" => BYTES,
      "description" => "The hash of managed deployment content that has been
      uploaded to the domain's or standalone server's deployment content
      repository.",
      "min-length" => 20,
      "max-length" => 20,
      "nillable" => false
      }
```

```
    }
}
```

Подробнее о том, как узнать об операциях, которые предоставляет ресурс, приведено далее по тексту.

### 2.3.3 Атрибуты

Ресурсы управления предоставляют информацию об их состоянии в виде атрибутов. Атрибуты имеют строковое имя и значение типа org.jboss.dmr.ModelNode (или: для CLI - текстовое представление ModelNode; для HTTP API - представление ModelNode в формате JSON).

Атрибуты могут быть доступны только для чтения или для записи. Чтение и запись значений атрибутов выполняется с помощью глобальных операций чтения и записи атрибутов.

Операция чтения атрибута принимает единственный параметр «name», значение которого является именем атрибута. Например, для чтения атрибута «port» ресурса, привязанного к сокету, через интерфейс командной строки:

```
[standalone@localhost:9990 /] /socket-binding-group=standard-
sockets/socketbinding=https:read-attribute(name=port)
{
 "outcome" => "success",
 "result" => 8443
}
```

Если атрибут доступен для записи, для изменения его состояния используется операция записи атрибута. Операция принимает два параметра:

– `name` - имя атрибута;

– `value` - значение атрибута.

Например, чтобы прочитать атрибут «port» ресурса, привязанного к сокету, через интерфейс командной строки:

```
[standalone@localhost:9990 /] /socket-binding-group=standard-
sockets/socketbinding=https:write-attribute(name=port,value=8444)
{"outcome" => "success"}
```

Атрибуты могут иметь один из двух возможных типов хранения:

– CONFIGURATION - означает, что значение атрибута хранится в постоянной конфигурации, т.е. в файле «domain.xml», «host.xml» или «standalone.xml», из которого была считана конфигурация ресурса;

– RUNTIME - значение атрибута доступно только на работающем сервере; оно не сохраняется в постоянной конфигурации. Показатель (например, количество обработанных запросов) является типичным примером атрибута «RUNTIME».

Значения всех атрибутов, которые предоставляет ресурс, могут быть получены с помощью операции «read-resource», при этом параметру «include-runtime» присвоено значение «true». Например, из командной строки:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=default-
server/httplistener=
default:read-resource(include-runtime=true)
{
 "outcome" => "success",
 "result" => {
  "allow-encoded-slash" => false,
  "allow-equals-in-cookie-value" => false,
  "always-set-keep-alive" => true,
  "buffer-pipelined-data" => true,
  "buffer-pool" => "default",
  "bytes-received" => 0L,
  "bytes-sent" => 0L,
  "certificate-forwarding" => false,
  "decode-url" => true,
  "disallowed-methods" => ["TRACE"],
  "enable-http2" => false,
  "enabled" => true,
  "error-count" => 0L,
  "max-buffered-request-size" => 16384,
  "max-connections" => undefined,
  "max-cookies" => 200,
  "max-header-size" => 1048576,
  "max-headers" => 200,
  "max-parameters" => 1000,
  "max-post-size" => 10485760L,
  "max-processing-time" => 0L,
  "no-request-timeout" => undefined,
  "processing-time" => 0L,
  "proxy-address-forwarding" => false,
  "read-timeout" => undefined,
  "receive-buffer" => undefined,
  "record-request-start-time" => false,
  "redirect-socket" => "https",
  "request-count" => 0L,
  "request-parse-timeout" => undefined,
  "resolve-peer-address" => false,
  "send-buffer" => undefined,
  "socket-binding" => "http",
  "tcp-backlog" => undefined,
  "tcp-keep-alive" => undefined,
  "url-charset" => "UTF-8",
  "worker" => "default",
  "write-timeout" => undefined
 }
}
```

Не указывайте параметр «include-runtime» (или установите для него значение «false»), чтобы ограничить вывод теми атрибутами, значения которых хранятся в постоянной конфигурации:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=default-
server/httplistener=default:read-resource(include-runtime=false)
{
 "outcome" => "success",
 "result" => {
  "allow-encoded-slash" => false,
  "allow-equals-in-cookie-value" => false,
  "always-set-keep-alive" => true,
  "buffer-pipelined-data" => true,
  "buffer-pool" => "default",
  "certificate-forwarding" => false,
  "decode-url" => true,
  "disallowed-methods" => ["TRACE"],
  "enable-http2" => false,
  "enabled" => true,
  "max-buffered-request-size" => 16384,
  "max-connections" => undefined,
  "max-cookies" => 200,
  "max-header-size" => 1048576,
  "max-headers" => 200,
  "max-parameters" => 1000,
  "max-post-size" => 10485760L,
  "no-request-timeout" => undefined,
  "proxy-address-forwarding" => false,
  "read-timeout" => undefined,
  "receive-buffer" => undefined,
  "record-request-start-time" => false,
  "redirect-socket" => "https",
  "request-parse-timeout" => undefined,
  "resolve-peer-address" => false,
  "send-buffer" => undefined,
  "socket-binding" => "http",
  "tcp-backlog" => undefined,
  "tcp-keep-alive" => undefined,
  "url-charset" => "UTF-8",
  "worker" => "default",
  "write-timeout" => undefined
 }
}
```

Далее по тексту приведено подробная информация об атрибутах, предоставляемых конкретным ресурсом.

Переопределите значение атрибута с помощью переменной окружения.

Можно переопределить значение любого простого атрибута, предоставив переменной окружения имя, соответствующее атрибуту (и его ресурсу).

**Внимание:** сложные атрибуты (тип которых задан как LIST, OBJECT или PROPERTY) не могут быть переопределены с помощью переменной окружения.

Если существует переменная среды с таким именем, ресурс управления будет использовать значение этой переменной среды, когда ресурс управления проверит и установит значение атрибута. Это происходит до того, как значение атрибута будет разрешено (если оно содержит выражение) или исправлено.

**Внимание:** по умолчанию эта функция отключена. Чтобы включить ее, необходимо задать переменную среды «WILDBOSS PRO_OVERRIDING_ENV_VARS» (ее значение не имеет значения): экспортировать «WILDBOSS PRO_OVERRIDING_ENV_VARS=1».

Сопоставление адреса и атрибута ресурса с переменной среды.

Имя переменной среды основано на адресе ресурса и имени атрибута:

1) возьмите адрес ресурса (например, /subsystem=undertow/server=default-сервер/httplistener= по умолчанию).

/subsystem=undertow/server=default-server/http-listener=default;

2) удалите начальную косую черту (/)

subsystem=undertow/server=default-server/http-listener=default;

3) добавьте два символа подчеркивания (__) и название атрибута (например, прокси-адрес-переадресация).

subsystem=undertow/server=default-server/http-listener=default__proxy-address-forwarding;

4) замените все не буквенно-цифровые символы символом подчеркивания (_) и введите его в верхнем регистре.

SUBSYSTEM_UNDERTOW_SERVER_DEFAULT_SERVER_HTTP_LISTENER_DEFAULT__PROXY_ADDRESS_FORWARDING.

Если WildBoss Pro запускается с использованием этой переменной среды, значение атрибута «proxy-address-forwarding» в «/subsystem=undertow/server=default-server/http-listener=default» будет равно значению переменной среды:

```
$ WILDBOSS PRO_OVERRIDING_ENV_VARS=1 \
SUBSYSTEM_UNDERTOW_SERVER_DEFAULT_SERVER_HTTP_LISTENER_DEFAULT__PROXY_A
DDRESS_FORWARDI
NG=false \
   ./bin/standalone.sh
$ ./bin/jboss-cli.sh -c --command="/subsystem=undertow/server=default-
server/http
-listener=default:read-attribute(name=proxy-address-forwarding)"
{
   "outcome" => "success",
   "result" => "false"
```

**Внимание:** если значение атрибута определено из переменной среды, то при следующем сохранении конфигурации это значение из переменной среды будет сохранено. Пока какая-либо операция не инициирует такое сохранение файла конфигурации, файл конфигурации не будет отражать текущую текущую конфигурацию.

### 2.3.4 Дочерние ресурсы

Ресурсы управления могут поддерживать дочерние ресурсы. Типы дочерних ресурсов, которые поддерживает ресурс (например, «connector» для ресурса веб-подсистемы), можно получить, запросив описание ресурса (см. описания далее по тексту) или выполнив операцию чтения типов дочерних ресурсов. Как только будут известны допустимые дочерние типы, пользователь можете запросить имена всех дочерних типов данного типа, используя глобальную операцию «read-children-types». Операция принимает единственный параметр «child-type», значением которого является тип. Например, ресурс, представляющий группу привязки сокетов, имеет дочерние элементы. Чтобы найти тип этих дочерних элементов и имена ресурсов этого типа с помощью интерфейса командной строки, можно:

```
[standalone@localhost:9990 /] /socket-binding-group=standard-sockets:read-
childrentypes
{
   "outcome" => "success",
   "result" => ["socket-binding"]
}
[standalone@localhost:9990 /] /socket-binding-group=standard-sockets:read-
childrennames(
child-type=socket-binding)
{
   "outcome" => "success",
   "result" => [
    "http",
    "https",
    "jmx-connector-registry",
    "jmx-connector-server",
    "jndi",
    "remoting",
    "txn-recovery-environment",
    "txn-status-manager"
   ]
}
```

### 2.3.5 Описания

Все ресурсы предоставляют метаданные, описывающие их атрибуты, операции и дочерние типы. Сами эти метаданные получаются путем вызова одной или нескольких глобальных операций, поддерживаемых каждым ресурсом. Ранее по тексту приведены примеры операций с именами операций чтения, описанием операций чтения, типами операций чтения дочерних элементов и именами операций чтения дочерних элементов.

Операция чтения описания ресурса может использоваться для поиска подробных сведений об атрибутах и дочерних типах, связанных с ресурсом. Например, с помощью интерфейса командной строки:

```
[standalone@localhost:9990 /] /socket-binding-group=standard-sockets:read-
resourcedescription
{
 "outcome" => "success",
 "result" => {
  "description" => "Contains a list of socket configurations.",
  "head-comment-allowed" => true,
  "tail-comment-allowed" => false,
  "attributes" => {
   "name" => {
    "type" => STRING,
    "description" => "The name of the socket binding group.",
    "required" => true,
    "head-comment-allowed" => false,
    "tail-comment-allowed" => false,
    "access-type" => "read-only",
    "storage" => "configuration"
   },
  "default-interface" => {
   "type" => STRING,
   "description" => "Name of an interface that should be used as the
   interface for any sockets that do not explicitly declare one.",
   "required" => true,
   "head-comment-allowed" => false,
```

```
    "tail-comment-allowed" => false,
    "access-type" => "read-write",
    "storage" => "configuration"
   },
   "port-offset" => {
    "type" => INT,
    "description" => "Increment to apply to the base port values defined
    in the socket bindings to derive the runtime values to use on this
    server.",
    "required" => false,
    "head-comment-allowed" => true,
    "tail-comment-allowed" => false,
    "access-type" => "read-write",
    "storage" => "configuration"
   }
  },
  "operations" => {},
  "children" => {"socket-binding" => {
   "description" => "The individual socket configurtions.",
   "min-occurs" => 0,
   "model-description" => undefined
  }}
 }
}
```

Обратите внимание на «операции ⇒ }}» в приведенных выше выходных данных. Если бы команда включала параметр «{{operations» (т.е. «/socket-binding-group=standard-sockets:read-resourcedescription( operations=true)»), то выходные данные включали бы описание каждой операции, поддерживаемой ресурсом.

Смотрите раздел «Глобальные операции» для получения подробной информации о других параметрах, поддерживаемых операцией «read-resourcedescription» и всеми другими глобально доступными операциями.

### 2.3.6 Сравнение с JMX MBeans

Ресурсы управления WildBoss Pro концептуально очень похожи на Open MBeans. У них есть следующие основные отличия:

– ресурсы управления WildBoss Pro организованы в виде древовидной структуры. Порядок пар «ключ-значение» в адресе ресурса имеет значение, поскольку он определяет положение ресурса в дереве. Порядок ключевых свойств в имени объекта JMX не имеет значения;

– в открытых значениях атрибутов MBean значения параметров операции и возвращаемые значения операций должны быть либо одного из простых типов JDK (String, Boolean, Integer и т.д.), либо реализованы в интерфейсе «javax.management.openmbean. CompositeData» или «javax.management.openmbean.TabularData» в табличных данных. Значения атрибутов ресурсов управления WildBoss Pro, значения параметров работы и возвращаемые значения операций имеют тип «org.jboss.dmr.ModelNode».

### 2.3.7 Базовая структура деревьев ресурсов управления

Как отмечалось выше, ресурсы управления организованы в виде древовидной структуры. Структура дерева зависит от того, используете ли вы автономный сервер или управляемый домен.

#### 2.3.7.1 Автономный сервер

Структура дерева управляемых ресурсов довольно близка к структуре файла конфигурации «standalone.xml».

Корневой ресурс:

– `extension` - расширения, установленные на сервере;

– `path` - пути, доступные на сервере;

– `system-property` - системные свойства, заданные как часть конфигурации (т.е. не в командной строке);

– `core-service=management` - основные службы управления сервером;

– `core-service=service-container` - ресурс для JBoss MSC ServiceContainer, который лежит в основе AS;

– `subsystem` - подсистемы, установленные на сервере. Основной частью модели управления будут дочерние элементы типа subsystem;

– `interface` - конфигурации интерфейсов;

– `socket-binding-group` - центральный ресурс для привязок сокетов сервера:

`socket-binding` - индивидуальные конфигурации привязки сокетов;

– `deployment` - доступные варианты развертывания на сервере.

#### 2.3.7.2 Доступные варианты развертывания на сервере

В управляемом домене структура дерева управляемых ресурсов охватывает весь домен, охватывая как конфигурацию всего домена (например, что находится в «domain.xml»), конфигурацию конкретного хоста для каждого хоста (например, что находится в «host.xml»), так и ресурсы, предоставляемые каждым запущенным сервером приложений. Процессы хост-контроллера в управляемом домене предоставляют доступ ко всему или к части общего дерева ресурсов. Объем доступных ресурсов зависит от того, взаимодействует ли клиент управления с хост-контроллером, который выступает в качестве главного контроллера домена. Если контроллер является главным контроллером домена, то доступна часть дерева для каждого хоста. Если хост-контроллер является подчиненным для удаленного контроллера домена, то доступна только часть дерева, связанная с этим хостом.

Кореневой ресурс для всего домена. Постоянные настройки, связанные с этим ресурсом и его дочерним, за исключением лиц, принимающих типа, сохраняются в файле «domain.xml» на контроллер домена:

– `extension` - расширения, доступные в домене;

– `path` - пути, доступные по всему домену;

– `system-property` - системные свойства задаются как часть конфигурации (т.е. не в командной строке) и доступны во всем домене;

– `profile` - наборы конфигураций подсистем, которые могут быть назначены группам серверов:

– `subsystem` - настройка подсистем, входящих в состав профиля;

– `interface` - конфигурации интерфейсов;

– `socket-binding-group` - наборы конфигураций привязок сокетов, которые могут быть применены к группам серверов:

– `socket-binding` - индивидуальные конфигурации привязки сокетов;

– `deployment` - развертывания, доступные для назначения группам серверов;

– `deployment-overlay` - содержимое развертывания-наложения, доступное для наложения развертываний в группах серверов;

– `server-group` - конфигурации групп серверов;

– host - на отдельных хост-контроллеры. Каждый дочерний ресурс этого типа представляет собой корневой ресурс для конкретного хоста. Стойкие конфигурации, связанные с одним из этих ресурсов, или своим дочерним сохраняются в файл «host.xml» хоста:

– path - пути, доступные на каждом сервере хоста;

– system-property - системные свойства, устанавливаемые на каждом сервере хоста;

– core-service=management - основные службы управления хост-контроллером;

– interface - конфигурации интерфейса, которые применяются к хост-контроллеру или серверам на хосте;

– jvm - конфигурации JVM, которые могут быть применены при запуске серверов;

– server-config - конфигурация, описывающая, как хост-контроллер должен запускать сервер; какую конфигурацию группы серверов использовать, а также любые зависящие от сервера переопределения элементов, указанных в других ресурсах;

– server - корневой ресурс для работающего сервера. Ресурсы, приведенные здесь и далее, не сохраняются напрямую; ресурсы домена и хоста содержат постоянную конфигурацию, которая управляет сервером:

– extension - расширения, установленные на сервере;

– path - пути, доступные на сервере;

– system-property - системные свойства, заданные как часть конфигурации (т.е. не в командной строке);

– core-service=management - основные службы управления сервером;

– core-service=service-container - ресурс для JBoss MSC ServiceContainer, который лежит в основе AS;

– subsystem - подсистемы, установленные на сервере. Основной частью модели управления будут дочерние элементы типа subsystem;

– interface - конфигурации интерфейсов;

– ocket-binding-group - центральный ресурс для привязок сокетов сервера:

– socket-binding - индивидуальные конфигурации привязки сокетов;

– deployment - доступные варианты развертывания на сервере;

– deployment-overlay - доступные оверлеи на сервере.

# 3 Управление клиентами

WildBoss Pro предлагает три различных подхода к настройке серверов и управлению клиентами: веб-интерфейс, клиент командной строки и набор XML-файлов конфигурации. Независимо от выбранного подхода, конфигурация всегда синхронизируется в разных представлениях и, в конечном итоге, сохраняется в XML-файлы.

## 3.1 Веб-интерфейс управления

Веб-интерфейс - это приложение GWT, которое использует HTTP management API для настройки домена управления или автономного сервера.

### 3.1.1 Конечная точка управления HTTP

Конечная точка HTTP API является точкой входа для клиентов управления, которые используют протокол HTTP для интеграции с уровнем управления. Он использует протокол в кодировке JSON и нетипизированный API в стиле RPC для описания и выполнения операций управления в управляемом домене или автономном сервере. Он используется веб-консолью, но предлагает возможности интеграции и для широкого круга других клиентов.

Конечная точка HTTP API находится либо на контроллере домена, либо на отдельном сервере. По умолчанию она работает через порт 9990.

```
<management-interfaces>
  [...]
   <http-interface http-authentication-factory="management-http-
   authentication">
    <http-upgrade enabled="true" sasl-authentication-
    factory="management-sasl-authentication"/>
    <socket-binding http="management-http"/>
   </http-interface>
<management-interfaces>
```

(Смотреть standalone/configuration/standalone.xml или domain/configuration/host.xml).

Конечная точка HTTP API обслуживает два разных контекста. Один из них предназначен для выполнения операций управления, а другой позволяет получить доступ к веб-интерфейсу:

– API домена: http://<host>:9990/management;

– Web-консоль: http://<host>:9990/console.

### 3.1.2 Доступ к Web-консоли

Web-консоль обслуживается через тот же порт, что и HTTP management API. Для доступа к ней необходимо указать в браузере на URL-адрес по умолчанию «http://<host>:9990/console».

### 3.1.3 Пользовательские HTTP-заголовки

Для ответов, возвращаемых из интерфейса управления HTTP, также можно определить пользовательские постоянные HTTP-заголовки, которые будут добавляться к любому ответу на основе сопоставления настроенного префикса с путем запроса.

В качестве примера далее по тексту приведен HTTP-заголовок X-Help, который указывает пользователям на правильное местоположение для получения помощи. В CLI

можно выполнить следующую операцию управления, чтобы активировать возврат этого заголовка во всех запросах.

```
[standalone@localhost:9990 /] /core-service=management/management-
interface=httpinterface: \
  write-attribute(name=constant-headers, value=[{path="/", \
  headers=[{name="X-Help", value="WildBoss Pro.org"}]}])
```

Ответы на все запросы к интерфейсу управления HTTP теперь будут содержать заголовок X-Help со значением WildBoss Pro.org.

Результирующая конфигурация будет выглядеть следующим образом:

```
<management-interfaces>
 <http-interface http-authentication-factory="management-http-
 authentication">
  <http-upgrade enabled="true" sasl-authentication-factory=
  "management-sasl-authentication"/>
   <socket-binding http="management-http"/>
   <constant-headers>
    <header-mapping path="/">
     <header name="X-Help" value="WildBoss Pro.org"/>
    </header-mapping>
   </constant-headers>
 </http-interface>
</management-interfaces>
```

Приведенный пример иллюстрирует добавление единого заголовка для всех запросов, соответствующих префиксу пути «/», т.е. для каждого запроса. Более сложные сопоставления могут быть определены путем указания сопоставления для более конкретного префикса пути, такого как «/management».

Если запрос соответствует нескольким сопоставлениям, таким как запрос в «/management», где были указаны сопоставления для «/» и «/management», заголовки из всех сопоставлений будут применены к соответствующему запросу.

В рамках одного сопоставления также можно определить несколько заголовков, которые должны быть установлены в соответствующем ответе.

Поскольку атрибут «constant-headers» установлен, будет выполнена проверка, чтобы убедиться, что в указанных HTTP заголовках используются только разрешенные символы, как указано в RFC спецификации HTTP.

Кроме того, поскольку они имеют специальную обработку в интерфейсе управления, переопределение следующих заголовков запрещено, и попытки установить их приведут к появлению сообщения об ошибке:

– «Connection» - соединение;

– «Content-Length» - длина содержимого;

– «Content-Type» - тип содержимого;

– «Date» - дата;

– «Transfer-Encoding» - передача-кодиторание.

Настроенные заголовки устанавливаются в самом конце обработки запроса непосредственно перед возвращением ответа клиенту, это будет означать, что любой из настроенных заголовков будет переопределять те же заголовки, установленные соответствующей конечной точкой.

## 3.2 Интерфейс командной строки

Интерфейс командной строки (CLI) - это инструмент управления управляемым доменом или автономным сервером. Он позволяет пользователю подключаться к контроллеру домена или автономному серверу и выполнять операции управления, доступные с помощью нетипизированной модели управления.

### 3.2.1 Запуск командной строки (CLI)

В зависимости от операционной системы интерфейс командной строки запускается с помощью файла «jboss-cli.sh» или «jboss-cli.bat», расположенного в каталоге «bin» WildBoss Pro. Дополнительная информация о структуре каталогов по умолчанию, приведена в документе «Руководству по началу работы».

Первое, что нужно сделать после запуска CLI, - это подключиться к управляемому экземпляру WildBoss Pro. Это делается с помощью команды «connect», например:

```
./bin/jboss-cli.sh
You are disconnected at the moment. Type 'connect' to connect to the
server or 'help' for the list of supported commands.
[disconnected /]

[disconnected /] connect
[domain@localhost:9990 /]

[domain@localhost:9990 /] quit
Closed connection to localhost:9990
```

«Localhost:9990» - это комбинация хоста и порта по умолчанию для CLI-клиента WildBoss Pro.

Хост и порт сервера могут быть указаны в качестве необязательных параметров, если сервер не прослушивает «localhost:9990».

```
./bin/jboss-cli.sh
You are disconnected at the moment. Type 'connect' to connect to the
server
[disconnected /] connect 192.168.0.10:9990
Connected to standalone controller at 192.168.0.1:9990
```

Значение «:9990» не требуется, так как интерфейс командной строки по умолчанию будет использовать порт 9990. Порт необходимо указать, если сервер прослушивает какой-либо другой порт.

Чтобы завершить сеанс, введите «quit».

**Внимание:** скрипт «jboss-cli» принимает параметр «--connect: ./jboss-cli.sh connect».

Параметр «--controller» может использоваться для указания хоста и порта сервера: «./jboss-cli.sh --connect --controller=192.168.0.1:9990».

Также доступна помощь.

Чтобы перечислить набор команд, которые в данный момент доступны в текущем контексте, используйте опцию «--commands» (в следующих примерах не показан полный набор команд CLI, в вашем запущенном экземпляре CLI может быть доступно больше и/или другие команды).

```
[domain@localhost:9990 /] help --commands
Commands available in the current context:
batch connection-factory deployment-overlay if
patch reload try
cd connection-info echo jdbc-driver-info
pwd rollout-plan undeploy
clear data-source echo-dmr jms-queue
quit run-batch unset
command deploy help jms-topic readattribute
set version
connect deployment-info history ls readoperation
shutdown xa-data-source
To read a description of a specific command execute 'help <command name>'.
```

Команда «help» может распечатать справку по любой команде или операции. Что касается операций, то описание операции оформлено в виде справки по команде (краткий обзор, описание и параметры). Некоторые команды (например, патч) предоставляют два уровня документации. Высокоуровневое описание самой команды и специальная справочная информация для каждого действия (например, «применить»). В справочной документации по каждой команде указывается, доступны ли эти два уровня или нет.

Чтобы ознакомиться с набором команд и операций необходимо перейти во вкладку «Завершение»:

```
help <TAB>
```

Отобразится список всех команд (включенных или нет).

Примеры:

– отобразить справку по команде исправления:

```
help patch
```

– отобразить справку о применении действия команды «patch»:

```
help patch apply
```

– отобразить описание операции добавления ресурса хранилища ключей elytron в формате справочного содержимого по команде:

```
help /subsystem=elytron/key-store=?:add
```

### 3.2.2 Навигация с помощью клавиатуры

Чтобы эффективно редактировать команды, интерфейс командной строки позволяет вам перемещаться по словам и символам команды с помощью клавиатуры.

**Внимание**: часть этой навигации зависит от платформы.

Перейти влево (назад) на одно слово:

– *Alt+B* : Linux, Solaris, HP-UX, Windows;

– *Ctrl+LeftArrow*: Linux, Solaris, HP-UX;
– *Alt+LeftArrow*: Mac OSX.
Идти вправо (вперед) одним словом:
– *Alt+F* : Linux, Solaris, HP-UX, Windows;
– *Ctrl+RightArrow*: Linux, Solaris, HP-UX;
– *Alt+RightArrow*: Mac OSX.
Перейдите к началу строки:
– *Ctrl+A*: All supported platforms;
– *HOME*: Linux, Solaris, HP-UX, Windows.
Пройдите до конца строки:
– *Ctrl+E*: All supported platforms;
– *END*: Linux, Solaris, HP-UX, Windows.
Перейти влево (назад) на один символ:
– *Ctrl+B* or *LeftArrow*: All supported platforms.
Перейти вправо (вперед) на один символ:
– *Ctrl+F* or *RightArrow*: All supported platforms.

### 3.2.3 Неинтерактивный режим

Интерфейс командной строки также может быть запущен в неинтерактивном режиме для поддержки сценариев и других типов командной строки или пакетной обработки. Аргументы «command» и «commands» могут использоваться для передачи команды или списка команд для выполнения. Кроме того, поддерживается аргумент «file», который позволяет выполнять команды CLI из текстового файла.

Например, следующая команда может быть использована для получения списка всех текущих развертываний:

```
$ ./bin/jboss-cli.sh --connect --commands=ls\ deployment
sample.war
business.jar
```

Выходные данные могут быть объединены с другими командами оболочки для дальнейшей обработки, например, для определения того, что именно файлы «war» развернуты:

```
$ ./bin/jboss-cli.sh --connect --commands=ls\ deployment | grep
war
sample.war
```

Чтобы сопоставить команду с ее выводом, вы можете указать параметр «echo-command» (или добавить элемент XML <echo-command> в файл конфигурации CLI), чтобы заставить CLI включать в вывод команду «prompt» + опции. Если эта опция включена, любая выполненная команда будет добавлена в выходные данные.

### 3.2.4 Время ожидания команды

По умолчанию выполнение команд и операций CLI не ограничено по времени. Это означает, что, если команда никогда не завершит свое выполнение, то процесс CLI будет зависать и перестанет отвечать на запросы. Чтобы защитить CLI от такого поведения, можно установить тайм-аут выполнения команды.

#### 3.2.4.1 Поведение по истечению времени ожидания команды

В интерактивном режиме при истечении времени ожидания отображается сообщение об ошибке, после чего в консоли становится доступным приглашение для ввода новых команд. В неинтерактивном режиме (выполнение скрипта или списка команд), когда наступает тайм-аут, генерируется исключение и выполнение CLI останавливается. В обоих режимах (интерактивном и неинтерактивном) при истечении времени ожидания интерфейс командной строки приложит все усилия, чтобы отменить соответствующие действия на стороне сервера.

#### 3.2.4.2 Настройка времени ожидания команды

Добавьте XML-элемент <время ожидания команды>\{количество секунд}</время ожидания команды> в XML -файл конфигурации CLI.

Добавьте параметр «время ожидания команды=\{количество секунд}» в командную строку CLI. Это переопределит любое значение, заданное в файле конфигурации XML.

#### 3.2.4.3 Управление тайм-аутом команды

Как только интерфейс командной строки запущен, тайм-аут можно настроить, чтобы он соответствовал выполняемым командам. Например, для пакетной команды потребуется более длительный тайм-аут, чем для не пакетной. Команда «command-timeout» позволяет получать, устанавливать и сбрасывать тайм-аут команды.

#### 3.2.4.4 Получение тайм-аута команды

Команда «command-timeout get» отображает текущее время ожидания в секундах. Значение времени ожидания, равное «0», означает, что время ожидания не истекло.

```
[standalone@localhost:9990 /] command-timeout get
0
```

#### 3.2.4.5 Установка времени ожидания команды

Команда «command-timeout set» обновляет значение тайм-аута до нескольких секунд. Если тайм-аут был задан с помощью конфигурации (XML-файла или опции), он переопределяется действием «set».

```
[standalone@localhost:9990 /] command-timeout set 10
```

#### 3.2.4.6 Сброс времени ожидания команды

Команда «command-timeout reset \{config|default}» позволяет установить время ожидания равным его конфигурационному значению (XML-файл или опция) или значению по умолчанию (0 секунд). Если значение конфигурации не задано, при возврате к значению конфигурации тайм-аут устанавливается на значение по умолчанию (0 секунд).

```
[standalone@localhost:9990 /] command-timeout reset config
[standalone@localhost:9990 /] command-timeout reset default
```

### 3.2.5 Безопасность встроенного интерфейса управления по умолчанию

Собственный интерфейс использует ту же конфигурацию безопасности, что и http-интерфейс, однако мы также поддерживаем механизм локальной аутентификации, который означает, что CLI может выполнять аутентификацию в локальном экземпляре WildBoss Pro, не запрашивая у пользователя имя пользователя и пароль. Этот механизм работает только в том случае, если пользователь, запускающий CLI, имеет доступ для чтения к автономной папке «/tmp/auth» или папке «domain/tmp/auth» в соответствующей установке WildBoss Pro - если локальный механизм не работает, CLI возвращает запрос на ввод имени пользователя и пароля для пользователя, настроенного по умолчанию безопасность HTTP-интерфейса.

Для установления CLI-соединения с удаленным сервером по умолчанию потребуются имя пользователя и пароль.

### 3.2.6 Запросы на выполнение операций

Запросы на выполнение операций обеспечивают низкоуровневое взаимодействие с моделью управления. Они отличаются от команд высокого уровня (например, «create-jms-queue») тем, что позволяют считывать и изменять конфигурацию сервера, как если бы вы редактировали файлы конфигурации XML напрямую. Конфигурация представлена в виде дерева адресуемых ресурсов, где каждый узел в дереве (он же ресурс) предлагает набор операций для выполнения.

Запрос на выполнение операции в основном состоит из трех частей: адреса, названия операции и необязательного набора параметров.

Формальной спецификацией для запроса на выполнение операции является:

```
[/node-type=node-name (/node-type=node-name)*] : operation-name
[( [parametername= parameter-value (,parameter-name=parameter-
value)*] )]
```

Например:

```
/subsystem=logging/root-logger=ROOT:change-root-log-
level(level=WARN)
```

Завершение работы на вкладке.

Завершение работы на вкладке поддерживается для всех команд и опций, т.е. для типов узлов и их названий, названий операций и параметров.

При завершении работы на вкладке обязательные параметры имеют название, заканчивающееся символом "*".

Это помогает определить, какие параметры необходимо задать, чтобы создать корректную операцию. Кроме того, в Tab-completion не предлагаются параметры, которые являются альтернативами параметрам, уже присутствующим в операции.

Например:

```
/deployment=myapp:add(<TAB>
! content* enabled runtime-name
```

Содержимое параметра является обязательным, и при заполнении оно обозначается символом "*".

```
/deployment=myapp:add-content(content=[{<TAB>
bytes* hash* input-stream-index* target-path* url*
```

Требуются байты, хэш, индекс входного потока и URL, но также и альтернативные значения (можно задать только одно из них). Как только один из этих параметров будет задан, остальные больше не будут предлагаться при завершении.

```
/deployment=myapp:add-content(content=[{url=myurl,<TAB>
/deployment=myapp:add-content(content=[{url=myurl,target-path
```

Аргумент «target-path» автоматически включается в команду.

**Внимание:** также рассматривается возможность добавления псевдонимов, которые будут менее подробными для пользователя и будут преобразованы в соответствующие запросы на выполнение операций в фоновом режиме.

Пробелы между разделителями в строках запроса на выполнение операции не имеют значения.

#### 3.2.6.1 Обращение к ресурсам

Запросы на выполнение операций не всегда могут содержать адресную часть или параметры.

Например:

```
:read-resource
```

В котором будут перечислены все типы узлов для текущего узла. Для синтаксического устранения неоднозначности между командами и операциями операциям требуется один из следующих префиксов:

– для выполнения операции с текущим узлом, например:

```
cd subsystem=logging
:read-resource(recursive="true")
```

– чтобы выполнить операцию с дочерним узлом текущего узла, например:

```
cd subsystem=logging
./root-logger=ROOT:change-root-log-level(level=WARN)
```

– чтобы выполнить операцию с корневым узлом, например:

```
/:read-resource
```

#### 3.2.6.2 Доступные типы операций и их описания

Типы операций можно разделить на общие операции, которые существуют на любом узле, и специфические операции, которые относятся к определенному конфигурационному ресурсу (т.е. подсистеме). Общими операциями являются:

– add;

– read-attribute;
– read-children-names;
– read-children-resources;
– read-children-types;
– read-operation-description;
– read-operation-names;
– read-resource;
– read-resource-description;
– remove;
– validate-address;
– write-attribute.

Для получения списка конкретных операций (например, операций, относящихся к подсистеме ведения журнала) вы всегда можете запросить саму модель. Например, для чтения операций, поддерживаемых ресурсом подсистемы ведения журнала на автономном сервере:

```
[[standalone@localhost:9990 /] /subsystem=logging:read-operation-names
{
   "outcome" => "success",
   "result" => [
     "add",
     "change-root-log-level",
     "read-attribute",
     "read-children-names",
     "read-children-resources",
     "read-children-types",
     "read-operation-description",
     "read-operation-names",
     "read-resource",
     "read-resource-description",
     "remove-root-logger",
     "root-logger-assign-handler",
     "root-logger-unassign-handler",
     "set-root-logger",
     "validate-address",
     "write-attribute"
  ]
}
```

Как вы можете видеть, ресурс ведения журнала предлагает четыре дополнительные операции, а именно: «root-logger-assignhandler», «root-logger-unassign-handler», «set-root-logger» и «remove-root-logger».

Дополнительную документацию о ресурсе или операции можно найти в описании:

```
[standalone@localhost:9990 /] /subsystem=logging:read-
operationdescription(name=change-root-log-level)
{
  "outcome" => "success",
  "result" => {
    "operation-name" => "change-root-log-level",
    "description" => "Change the root logger level.",
    "request-properties" => {"level" => {
      "type" => STRING,
      "description" => "The log level specifying which message levels
      will be logged by this logger.
```

```
                    Message levels lower than this value will be
                    discarded.",
    "required" => true
   }}
  }
}
```

Полная модель.

**Внимание:** чтобы увидеть полную модель, введите «:read-resource(recursive=true)».

### 3.2.7 История команд

Журнал команд (и запросов на выполнение) включен по умолчанию. Журнал хранится как в памяти, так и в файле на диске, т.е. сохраняется между сеансами работы с командной строкой. Файл истории называется «.jboss-cli-history» и автоматически создается в домашнем каталоге пользователя. При запуске интерфейса командной строки этот файл считывается, и его содержимое инициализируется в истории в памяти.

**Внимание:** находясь в сеансе работы с командной строкой, вы можете использовать клавиши со стрелками для перемещения вперед и назад по истории команд и операций.

Чтобы управлять историей, вы можете использовать команду «history». Если она выполняется без каких-либо аргументов, она распечатает все записанные команды и операции (вплоть до заданного максимального значения, которое по умолчанию равно 500) из истории в памяти.

История поддерживает три необязательных аргумента:

– *disable* - отключит расширение истории (но не очистит ранее записанную историю);

– *enabled* - повторно включит расширение хронологии (начиная с последней записанной команды до того, как расширение хронологии было отключено);

– *clear* - очистит историю в памяти (но не файловую).

### 3.2.8 Вывод в формате JSON и DMR

По умолчанию CLI выводит результаты операций, используя текстовый синтаксис DMR. Есть два способа заставить CLI отображать JSON:

– *«output-json»* параметр при запуске CLI;

– *«output-json»* XML-элемент добавлен в файл конфигурации «jboss-cli.xml».

### 3.2.9 Цветопередача

Интерфейс командной строки отображает результаты выполнения команд и приглашение в цветном виде. Чтобы отключить это, есть два возможных способа отключить это:

– *«no-color-output»* отключит вывод цвета;

– измените значение «*enabled*» на значение false в «jboss-cli.xml».

Блок «color-output» используется для настройки цветов шести базовых элементов, которые его поддерживают:

– выходные сообщения: ошибка, предупреждение и успех;

– необходимые параметры конфигурации при использовании функции автозаполнения;

– цвет приглашения по умолчанию;

– цвет приглашения при использовании пакета и любой из команд рабочего процесса, «if», «for» и «try».

```
<color-output>
  <enabled>true</enabled>
  <error-color>red</error-color>
  <warn-color>yellow</warn-color>
  <success-color>default</success-color>
  <required-color>magenta</required-color>
  <workflow-color>green</workflow-color>
  <prompt-color>blue</prompt-color>
</color-output>
```

Существует восемь доступных цветов:

– black - черный;

– blue - синий;

– cyan - голубой;

– green - зеленый;

– magenta - пурпурный;

– red - красный;

– white - белый;

– yellow - желтый.

Существует также возможность использования цвета по умолчанию, который является настроенным цветом переднего плана терминала.

### 3.2.10 Вывод данных для подкачки и поиска

В интерактивном режиме, когда длина отображаемого содержимого превышает высоту терминала, оно разбивается на страницы. Вы можете перемещаться по содержимому с помощью следующих клавиш и действий мыши:

– пробел или «PAGE_DOWN»: прокрутите содержимое на одну страницу вниз;

– «\» или «PAGE_UP»: прокрутите содержимое на одну страницу вверх;

– «;» или стрелка вверх, или колесико мыши вверх: прокрутите содержимое на одну строку вверх;

– «ENTER», стрелка вниз или колесико мыши вниз: прокрутите содержимое на одну строку вниз;

– «HOME» или «g»: прокрутите страницу до начала содержимого.

**Внимание:** «HOME» поддерживается только для клавиатур, содержащих эту клавишу;

– «END» или «G»: прокрутите содержимое до конца.

**Внимание:** END поддерживается только для клавиатур, содержащих эту клавишу;

– «q» или «Q» или «ESC»: выход из режима подкачки.

**Внимание:** при достижении конца содержимого (с помощью ввода, пробела, ...) подкачка автоматически завершается.

Можно выполнять поиск текста при постраничном просмотре содержимого. Поиск осуществляется следующим образом ключи:

– «/» для отображения запроса, позволяющего ввести некоторый текст. Введите «return» для запуска поиска. Вы можете использовать стрелки «вверх/вниз» для поиска ранее введенного текста.

**Внимание:** история поиска не сохраняется при завершении процесса CLI;

– «n» для перехода к следующему совпадению, если таковое имеется. Если текст для поиска не был введен, используется последняя запись из истории поиска;

– «N» для перехода к предыдущему совпадению, если таковое имеется. Если текст для поиска не был введен, используется последняя запись из истории поиска.

Существует два возможных способа отключить подкачку выходных данных и записать весь вывод команд сразу:

– параметр командной строки «no-output-paging» отключает подкачку выходных данных;

– добавьте <*output-paging*>*false*<*output-paging*> в «jboss-cli.xml».

**Внимание:** в Windows поиск и обратная навигация поддерживаются только начиная с Windows 10 и Windows Server 2016.

**Внимание:** если процесс CLI отправит сигнал KILL(9) во время выполнения подкачки, терминал останется в альтернативном режиме. Это приведет к неожиданному поведению терминала (события отображения и мыши). Чтобы восстановить состояние терминала, вызовите: «tput rmcup».

### 3.2.11 Пакетная обработка

Пакетный режим позволяет группировать команды и операции и выполнять их как единое целое. Если хотя бы одна из команд или операций завершается ошибкой, все остальные успешно выполненные команды и операции в пакете отменяются.

Не все команды разрешены в пакете. Например, такие команды, как «cd», «ls», «help» и т.д., не разрешены в пакетном режиме, поскольку они не преобразуются в запросы на выполнение операций. В пакетном режиме разрешены только команды, которые преобразуются в запросы на выполнение операций. На самом деле пакет выполняется как составной запрос на выполнение операций.

Переход в пакетный режим осуществляется путем выполнения команды «batch».

```
[standalone@localhost:9990 /] batch
[standalone@localhost:9990 / #]
/subsystem=datasources/datasource="
java\:\/H2DS":enable
[standalone@localhost:9990 / #] /subsystem=messaging-
activemq/server=default/jmsqueue=
newQueue:add
```

Вы можете выполнить пакетный запуск с помощью команды «run-batch»:

```
[standalone@localhost:9990 / #] run-batch
The batch executed successfully.
```

Выйдите из режима пакетного редактирования без потери внесенных изменений:

```
[standalone@localhost:9990 / #] holdback-batch
[standalone@localhost:9990 /]
```

Затем снова активируйте его позже:

```
[standalone@localhost:9990 /] batch
Re-activated batch
#1 /subsystem=datasources/data-source=java:/H2DS:\/H2DS:enable
```

Также доступно несколько других примечательных пакетных команд (вкладка завершена, чтобы просмотреть список):

– *clear-batch*;

– *dit-batch-line* (например, *edit-batch line* 3 *create-jms-topic name=mytopic*);

– *remove-batch-line* (например, *remove-batch-line 3*);
– *move-batch-line* (например, *move-batch-line 3 1*);
– *discard-batch*.

### 3.2.12 Операторы

В CLI есть несколько операторов, которые похожи на операторы оболочки:
– «>» перенаправить вывод команды/операции в файл:

```
:read-resource > my-file.txt
```

– «>>» перенаправить вывод команды/операции и добавить его в конец файла:

```
:read-resource >> my-file.txt
```

– «|» чтобы перенаправить вывод команды/операции в команду grep:

```
:read-resource | grep undefined
```

## 3.3 Безопасность HTTP-интерфейса по умолчанию

Распространяемый WildBoss Pro защищен по умолчанию. Механизм защиты по умолчанию основан на использовании имени пользователя и пароля и использует HTTP-дайджест для процесса аутентификации.

Причина защиты сервера по умолчанию заключается в том, что в случае случайного доступа к интерфейсам управления по общедоступному IP-адресу для подключения требуется аутентификация - по этой причине в дистрибутиве нет пользователя по умолчанию.

Данные пользователя хранятся в файле свойств под названием «mgmt-users.properties» в разделах «standalone/configuration» и «domain/configuration» в зависимости от режима работы сервера, эти файлы содержат имя пользователя, а также предварительно подготовленный хэш имени пользователя, название области и пароль пользователя.

**Внимание:** хотя файлы свойств не содержат паролей в виде обычного текста, они все равно должны быть защищены, поскольку заранее подготовленные хэши могут быть использованы для получения доступа к любому серверу с той же областью, если один и тот же пользователь использовал один и тот же пароль.

Для работы с файлами и добавления пользователей предоставляется утилита «add-user.sh» и «add-user.bat» для добавления пользователей и генерации хэшей, чтобы добавить пользователя, вы должны запустить скрипт и следовать инструкциям process. [images/add-user.png].

Полная информация о программе добавления пользователя описана ниже, но для доступа к интерфейсу управления вам необходимо ввести следующие значения:

– тип пользователя - это будет «Управляющий пользователь» для выбора опции a;

– область - это должно совпадать с именем области, используемым в конфигурации, поэтому, если вы не изменили конфигурацию для использования другого имени области, оставьте это значение как «ManagementRealm»;

– имя пользователя - имя пользователя, которого вы добавляете;

– пароль - пароль пользователя.

Если проверка прошла успешно, предлагается подтвердить добавление пользователя, и файлы свойств будут обновлены.

На последний вопрос, поскольку это пользователь, который собирается получить доступ к консоли администратора, просто ответить «n» - эта опция будет описана позже для добавления подчиненных контроллеров хоста, которые проходят проверку подлинности на главном контроллере домена, но это более поздний раздел.

После добавления нового пользователя сервер должен быть перезапущен или должна быть выполнена операция загрузки ресурса «ManagementRealm» или «ApplicationRealm» в подсистеме «elytron», в зависимости от обстоятельств.

```
[standalone@localhost:9990 /] /subsystem=elytron/properties-
realm=ManagementRealm:load
{"outcome" => "success"}
```

### 3.4 Безопасность встроенного интерфейса по умолчанию

Собственный интерфейс использует ту же конфигурацию безопасности, что и http-интерфейс, однако мы также поддерживаем механизм локальной аутентификации, который означает, что CLI может выполнять аутентификацию в локальном экземпляре WildBoss Pro, не запрашивая у пользователя имя пользователя и пароль. Этот механизм работает только в том случае, если пользователь, запускающий CLI, имеет доступ для чтения к автономной папке/tmp/auth или папке domain/tmp/auth в соответствующей установке WildBoss Pro - если локальный механизм не работает, CLI снова запросит имя пользователя и пароль для пользователя, настроенного по умолчанию Безопасность HTTP-интерфейса.

Для установления CLI-соединения с удаленным сервером по умолчанию потребуется ввести имя пользователя и пароль.

### 3.5 Интерфейс командной строки

Интерфейс командной строки (CLI) - это инструмент управления управляемым доменом или автономным сервером. Он позволяет пользователю подключаться к контроллеру домена или автономному серверу и выполнять операции управления, доступные с помощью нетипизированной модели управления.

Подробную информацию о том, как использовать интерфейс командной строки, можно найти на странице «Интерфейса командной строки».

### 3.6 Конфигурационные файлы

WildBoss Pro сохраняет свою конфигурацию в централизованных XML-файлах конфигурации, по одному на сервер для автономных серверов и, для управляемых доменов, по одному на хост с дополнительной политикой для всего домена, контролируемой главным хостом. Эти файлы должны быть удобочитаемыми и редактируемыми пользователем.

**Внимание:** файлы конфигурации XML выступают в качестве центрального, авторитетного источника конфигурации. Любые изменения конфигурации, внесенные через веб-интерфейс или командную строку, сохраняются в файлах конфигурации XML. Если домен или автономный сервер отключен, то файлы конфигурации XML также можно редактировать вручную, и любые изменения будут учтены при следующем запуске домена или автономного сервера. Однако пользователям рекомендуется использовать веб-интерфейс или CLI, предпочитая внесение изменений в файлы конфигурации в автономном режиме. Внешние изменения, внесенные в файлы конфигурации во время выполнения процессов, не будут обнаружены и могут быть перезаписаны.

#### 3.6.1 Файл конфигурации автономного сервера

Конфигурацию XML для автономного сервера можно найти в каталоге «standalone/configuration». Конфигурационный файл по умолчанию

«standalone/configuration/standalone.xml». Каталог «standalone/configuration» содержит ряд других стандартных конфигурационных файлов, например «standalone-full.xml», «standalone-ha.xml» и «standalone-full-ha.xml», каждый из которых аналогичен файлу по умолчанию «standalone.xml», но включает дополнительные подсистемы, отсутствующие в конфигурации по умолчанию. Если вы предпочитаете использовать один из этих файлов в качестве конфигурации вашего сервера, вы можете указать его с помощью аргумента командной строки «*[line-through]*c** or *-server-config*»:

– bin/standalone.sh -c=standalone-full.xml;

– bin/standalone.sh --server-config=standalone-ha.xml.

### 3.6.2 Файлы конфигурации управляемого домена

В управляемом домене XML-файлы находятся в каталоге domain/configuration. Существует два типа файлов конфигурации – по одному для каждого хоста и один файл для всего домена, управляемый главным хостом, то есть контроллером домена. (Подробнее о типах процессов в управляемом домене в разделе «Режимы работы»)

Конфигурация для конкретного хоста – host.xml.

Когда вы запускаете процесс управляемого домена, запускается экземпляр контроллера хоста, и он анализирует свой собственный файл конфигурации, чтобы определить свою собственную конфигурацию, способ интеграции с остальным доменом, любые значения параметров для хоста в конфигурации всего домена (например, IP-адреса) и какие серверы он должен использовать. должен запуститься. Эта информация содержится в файле конфигурации конкретного хоста, версия которого по умолчанию равна «domain/configuration/host.xml».

У каждого хоста будет свой собственный вариант «host.xml» с настройками, соответствующими его роли в домене. WildBoss Pro поставляется с тремя стандартными вариантами:

– *host-master.xml* - конфигурация, в которой указан хост Контроллер должен стать главным, он же Контроллер домена. Серверы не будут запускаться этим хост-контроллером, что является рекомендуемой настройкой для рабочего главного сервера;

– *host-slave.xml* - конфигурация, в которой указан хост Контроллер не должен становиться ведущим, а вместо этого должен регистрироваться на удаленном главном сервере и управляться им. Эта конфигурация запускает серверы, хотя пользователь, вероятно , захочет изменить количество запущенных серверов и группы серверов, к которым они принадлежат;

– *host.xml* - конфигурация хоста по умолчанию, разработанная специально для упрощения экспериментов с управляемым доменом. В этой конфигурации контроллер хоста должен стать главным, он же контроллер домена, но при этом также запускается несколько серверов.

Какую конфигурацию конкретного хоста следует использовать, можно определить с помощью аргумента командной строки «_ --host-config_»:

```
$ bin/domain.sh --host-config=host-master.xml
```

#### 3.6.2.1 Конфигурация для всего домена – domain.xml

Как только хост-контроллер обработает свою конфигурацию, специфичную для хоста, он узнает, является ли она сконфигурирован для выполнения функций главного контроллера домена. Если это так, то он должен проанализировать файл конфигурации всего домена, который по умолчанию находится по адресу «domain/configuration/domain.xml». Этот файл содержит основную часть настроек, которые должны быть применены к серверам в домене при их запуске – среди прочего, какие подсистемы они должны запускать, с какими настройками, какие сокеты следует использовать и какие развертывания следует выполнять.

Какую конфигурацию для всего домена следует использовать, можно определить с помощью аргумента командной строки «_ --domain-config_»:

```
$ bin/domain.sh --domain-config=domain-production.xml
```

Этот аргумент применим только к хостам, настроенным на выполнение функций ведущего.

Подчиненный хост-контроллер обычно не анализирует файл конфигурации всего домена. Подчиненный сервер получает конфигурацию всего домена от удаленного главного контроллера домена при регистрации на нем. Подчиненный сервер также не будет сохранять изменения в файле «domain.xml», если он присутствует в файловой системе. По этой причине рекомендуется, чтобы в файловой системе хостов, которые будут работать только как подчиненные, не сохранялось никаких «domain.xml».

Подчиненное устройство можно настроить так, чтобы оно сохраняло локальную копию конфигурации всего домена и затем использовало ее при загрузке (в случае, если ведущее устройство недоступно). Смотрите раздел «--backup» и «--cached-dc» в разделе «Параметры командной строки».

# 4 Интерфейсы и порты

## 4.1 Объявления интерфейса

WildBoss Pro использует ссылки на именованные интерфейсы во всей конфигурации. Сетевой интерфейс объявляется путем указания логического имени и критериев выбора физического интерфейса:

```
[standalone@localhost:9990 /] :read-children-names(child-
type=interface)
{
   "outcome" => "success",
   "result" => [
   "management",
   "public"
   ]
}
```

Это означает, что сервер, о котором идет речь, объявляет два интерфейса: один называется «*public*» («управление»), другой – «*management*» («общедоступный»). Интерфейс «*public*» используется для всех компонентов и служб, которые требуются на уровне управления (т.е. для конечной точки управления HTTP). Привязка к «*management*» интерфейсу

используется для любого сетевого взаимодействия, связанного с приложениями (например, Веб, обмен сообщениями и т.д.).

В этих именах нет ничего особенного; интерфейсы могут быть объявлены с любым именем. Затем другие разделы конфигурации могут ссылаться на эти интерфейсы по их логическому имени, вместо того чтобы включать полную информацию об интерфейсе (которая на серверах в домене управления может отличаться на разных компьютерах).

Тот самый «domain.xml», «host.xml» и «standalone.xml» - все конфигурационные файлы содержат раздел, в котором могут быть объявлены интерфейсы. Если мы взглянем на XML-декларацию, то увидим критерии выбора. Критерии могут быть одного из двух типов: либо один элемент, указывающий на то, что интерфейс должен быть привязан к подстановочному адресу, либо набор из одной или нескольких характеристик, которыми должен обладать интерфейс или адрес, чтобы они соответствовали действительности. Критериями выбора в этом примере являются конкретные IP -адреса для каждого интерфейса:

```
<interfaces>
  <interface name="management">
    <inet-address value="127.0.0.1"/>
  </interface>
  <interface name="public">
    <inet-address value="127.0.0.1"/>
  </interface>
</interfaces>
```

Некоторые другие примеры:

```
<interface name="global">
  <!-- Use the wildcard address -->
  <any-address/>
</interface>

<interface name="external">
```

```
  <nic name="eth0"/>
 </interface>

 <interface name="default">
  <!-- Match any interface/address on the right subnet if it's
  up, supports multicast and isn't point-to-point -->
  <subnet-match value="192.168.0.0/16"/>
  <up/>
  <multicast/>
  <not>
    <point-to-point/>
  </not>
 </interface>
```

**Внимание:** элемент конфигурации интерфейса используется для предоставления единого адреса InetAddress частям сервера, которые ссылаются на этот интерфейс. Если в результате выполнения критериев выбора, указанных для элемента интерфейса, более чем один адрес удовлетворяет этим критериям, то будет выдано предупреждение и будет выбран и использован только один адрес. Предпочтение будет отдаваться сетевым интерфейсам, которые подключены, не имеют обратной связи и не являются двухточечными.

### 4.1.1 Аргумент командной строки -b

WildBoss Pro поддерживает использование аргумента командной строки «-b» для указания адреса, назначаемого интерфейсам.

Дополнительные сведения см. в разделе «Управление адресом привязки с помощью «- b»».

## 4.2 Группы привязки сокетов

Конфигурация сокетов в WildBoss Pro работает аналогично объявлениям интерфейсов. Сокеты объявляются с использованием логического имени, по которому на них будут ссылаться во всей конфигурации.

Объявления сокетов группируются под определенным именем. Это позволяет вам легко ссылаться на определенную группу привязки сокетов при настройке групп серверов в управляемом домене.

Группы привязки сокетов ссылаются на интерфейс по его логическому имени:

```
<socket-binding-group name="standard-sockets" default-
interface="public">
  <socket-binding name="management-http" interface="management" port=
  "${jboss.management.http.port:9990}"/>
  <socket-binding name="management-https" interface="management" port=
  "${jboss.management.https.port:9993}"/>
  <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
  <socket-binding name="http" port="${jboss.http.port:8080}"/>
  <socket-binding name="https" port="${jboss.https.port:8443}"/>
  <socket-binding name="txn-recovery-environment" port="4712"/>
  <socket-binding name="txn-status-manager" port="4713"/>
</socket-binding-group>
```

Привязка к сокету содержит следующую информацию:

– «name» - логическое имя конфигурации сокета, которое должно использоваться в другом месте конфигурации;

– «port» - базовый порт, к которому должен быть привязан сокет, основанный на данной конфигурации (обратите внимание, что серверы могут быть настроены таким образом, чтобы

переопределять это базовое значение, применяя увеличение или уменьшение ко всем значениям порта);

– «interface» (необязательно) - логическое имя (см. «Объявления интерфейсов» выше) интерфейса, к которому должен быть привязан сокет, основанный на данной конфигурации. Если оно не определено, будет использоваться значение атрибута «default-interface» из группы привязки сокета, содержащей это имя;

– «multicast-address» (необязательно) - если сокет будет использоваться для многоадресной рассылки, то используемый адрес многоадресной рассылки;

– «multicast-port» (необязательно) - если сокет будет использоваться для многоадресной рассылки, то используемый порт многоадресной рассылки;

– «fixed-port» (необязательно, по умолчанию используется значение «false») - если значение равно «true», то объявляется, что значение «port» всегда должно использоваться для сокета и не должно быть переопределено путем применения увеличения или уменьшения.

## 4.3 IPv4 против IPv6

WildBoss Pro поддерживает использование как IPv4, так и IPv6 адресов. По умолчанию WildBoss Pro настроен для использования в сети IPv4, поэтому, если вы работаете в сети IPv4, никаких изменений не требуется. Если вам необходимо работать в сети IPv6, необходимые изменения минимальны и включают в себя изменение стека JVM и настроек адресов, а также настройку любых значений IP-адресов интерфейса, указанных в 87 конфигурации («standalone.xml» или «domain.xml»).

### 4.3.1 Предпочтение стека и адреса

Системные свойства «java.net.preferIPv4Stack» и «java.net.preferIPv6Addresses» используются для настройки виртуальной машины JAVA для использования с адресами IPv4 или IPv6. В WildBoss Pro для запуска с использованием IPv4-адресов вам необходимо указать «java.net.preferIPv4Stack=true»; для запуска с IPv6-адресами вам необходимо указать «java.net.preferIPv4Stack=false» (значение JVM по умолчанию) и «java.net.preferIPv6Addresses=true». Последнее гарантирует, что при любом преобразовании имени хоста в IP-адрес всегда будут возвращаться варианты IPv6-адресов.

Эти системные свойства удобно задаются переменной окружения JAVA_OPTS, определенной в файле «standalone.conf» (или «domain.conf»). Например, чтобы изменить настройки стека IP с IPv4 по умолчанию на IPv6, отредактируйте файл «standalone.conf» (или «domain.conf») и измените его настройки IPv4 по умолчанию:

```
if [ "x$JAVA_OPTS" = "x" ]; then
JAVA_OPTS=" ... -Djava.net.preferIPv4Stack=true ..."
...
```

К подходящей настройке IPv6:

```
if [ "x$JAVA_OPTS" = "x" ]; then
JAVA_OPTS=" ... -Djava.net.preferIPv4Stack=false
-Djava.net.preferIPv6Addresses=true ..."
...
```

### 4.3.2 Литералы IP-адресов

Чтобы изменить литералы IP-адресов, указанные в «standalone.xml» (или «domain.xml»), сначала просмотрите объявления интерфейса и убедитесь, что в качестве значений интерфейса используются действительные адреса IPv6. Например, чтобы изменить

конфигурацию по умолчанию, в которой интерфейс обратной связи используется в качестве основного интерфейса, измените адрес обратной связи с IPv4:

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
</interfaces>
```

К обратному адресу IPv6:

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:[::1]}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:[::1]}"/>
  </interface>
</interfaces>
```

Обратите внимание, что при включении литералов IPv6-адреса в выражение для подстановки используются квадратные скобки, заключающие литерал IP-адреса во избежание двусмысленности. Это соответствует соглашению об использовании литералов IPv6 в URL-адресах.

Помимо внесения таких изменений в определения интерфейса, вам также следует проверить остальную часть вашего конфигурационного файла и при необходимости изменить литералы IP-адресов с IPv4 на IPv6.

# 5 Административная безопасность

## 5.1 Утилита добавления пользователя

Для использования с конфигурацией по умолчанию мы предоставляем утилиту «add-user», которая может использоваться для управления файлами свойств для областей по умолчанию, используемых для хранения пользователей и их ролей.

Утилиту добавления пользователя можно использовать для управления как пользователями в ManagementRealm, так и пользователями в ApplicationRealm, внесенные изменения применяются к файлу свойств, используемому как для режима домена, так и для автономного режима.

**Внимание:** после того, как вы установили свой сервер приложений и решили, будете ли вы работать в автономном режиме или в режиме домена, вы можете удалить родительскую папку для режима, который вы не используете, тогда утилита добавления пользователя будет управлять файлом свойств только для используемого режима.

Утилита добавления пользователя - это утилита командной строки, однако ее можно запускать как в интерактивном, так и в неинтерактивном режиме. В зависимости от вашей платформы скриптом для запуска утилиты добавления пользователя является «adduser.sh» или «add-user.bat», которые можно найти в «\{ jboss.home}/bin».

В этом руководстве приведено несколько примеров использования этой утилиты для выполнения наиболее распространенных задач.

### 5.1.1 Добавление пользователя

Основной целью этой утилиты является добавление пользователей в файлы свойств. Имена пользователей могут содержать только следующие символы в любом количестве и в любом порядке:

– буквенно-цифровые символы (a-z, A-Z, 0-9);
– тире (-), точки (.), запятые (,), в (@);
– экранированная обратная косая черта ( \\ );
– экранированные значения (\=).

#### 5.1.1.1 Управляющий пользователь

**Внимание:** имя области по умолчанию для пользователей «management» - ManagementRealm, и когда утилита запросит имя области, просто примите значение по умолчанию, если только вы не переключились на другую область.

1) Интерактивный режим.

[images/add-mgmt-user-interactive.png]

Здесь мы добавили нового управляющего пользователя под названием AdminUser, так как вы можете видеть, что на некоторые вопросы предлагаются ответы по умолчанию, поэтому вы можете просто нажать enter, не повторяя значение по умолчанию. А пока просто ответьте "нет" на последний вопрос. Добавление пользователей, которые будут использоваться процессами, более подробно описано в главе "Управление доменом".

2) Неинтерактивный режим.

Для добавления пользователя в неинтерактивном режиме можно использовать команду «./add-user.sh {имя пользователя} {пароль}».

[images/add-mgmt-user-non-interactive.png]

**Внимание:** если вы добавляете пользователей, используя этот подход, существует риск того, что любой другой пользователь, который может просматривать список запущенных процессов, может увидеть аргументы, включая пароль добавляемого пользователя, также существует риск того, что комбинация имени пользователя и пароля будет кэширована в файле истории оболочки, в которой вы в данный момент находитесь.

### 5.1.2 Обновление пользователя

С помощью утилиты добавления пользователей также можно обновить существующих пользователей, в интерактивном режиме вам будет предложено подтвердить, является ли это вашим намерением.

#### 5.1.2.1 Управляющий пользователь

1) Интерактивный режим.
[images/update-mgmt-user-interactive.png]
2) Неинтерактивный режим.
В неинтерактивном режиме, если пользователь уже существует, обновление происходит автоматически, без запроса подтверждения.

#### 5.1.2.2 Пользователь приложения

1) Интерактивный режим.
[images/update-app-user-interactive.png]
**Внимание:** при обновлении роли пользователя вам нужно будет повторно ввести список ролей, назначенных пользователю.
2) Неинтерактивный режим.
В неинтерактивном режиме, если пользователь уже существует, обновление происходит автоматически, без запроса подтверждения.

### 5.1.3 Вклад сообщества

В утилиту добавления пользователей еще можно добавить несколько функций, таких как удаление пользователей или добавление пользователей приложения с ролями в неинтерактивном режиме, если вы заинтересованы в том, чтобы внести свой вклад в Разработка с помощью WildBoss Pro утилита добавления пользователей - это хорошее место для начала, поскольку это автономная утилита, однако она является частью сборки AS, поэтому вы можете ознакомиться с процессами разработки AS , не вникая непосредственно во внутренние устройства сервера приложений.

## 5.2 Авторизация управленческих действий на основе ролей Контроль доступа

WildBoss Pro представляет схему управления доступом на основе ролей, которая позволяет разным пользователям с правами администратора иметь разные наборы разрешений на чтение и обновление частей дерева управления. Это заменяет простую схему разрешений, где любой, кто сможет успешно пройти аутентификацию в области безопасности управления, получит все разрешения.

### 5.2.1 Поставщики услуг по контролю доступа

WildBoss Pro поставляется с двумя «провайдерами» контроля доступа: «простым» провайдером и «rbac» провайдером. По умолчанию используется «простой» провайдер, который предоставляет схему разрешений при которой любой прошедший проверку подлинности администратор имеет все разрешения. Поставщик «rbac» предоставляет более детализированную схему разрешений, которая является предметом рассмотрения в этом разделе.

Конфигурация контроля доступа включена в раздел «Управление» автономного сервера «standalone.xml» или в новый раздел «управление» в управляемом домене «domain.xml». Политика контроля доступа настраивается централизованно в управляемом домене.

```
<management>
...
  <access-control provider="simple">
     <role-mapping>
     <role name="SuperUser">
        <include>
           <user name="$local"/>
        </include>
     </role>
     </role-mapping>
  </access-control>
</management>
```

Как вы можете видеть, по умолчанию для провайдера установлено значение «простой». Для «простого» провайдера вложенный раздел «сопоставление ролей» на самом деле не имеет значения. Это необходимо для того, чтобы гарантировать, что если атрибут «provider» будет изменен на «rbac», то по крайней мере одному пользователю будет присвоена роль, которая сможет продолжать администрировать систему. Это сопоставление по умолчанию присваивает имя пользователя «$local» роли «RBAC», которая предоставляет все разрешения - роли «Суперпользователь». Имя пользователя «$local» - это имя, которое будет присвоено администратору, если он или она использует интерфейс командной строки в той же системе, что и экземпляр WildBoss Pro, и включена схема «локальной» аутентификации.

### 5.2.2 Обзор поставщика RBAC

Схема управления доступом, реализуемая провайдером «rbac», основана на семи стандартных ролях. Роль - это именованный набор разрешений для выполнения одного из действий: обращения (т.е. поиска) к ресурсу управления, его чтения или изменения. К различным ролям применяются ограничения на их разрешения, которые используются для определения того, будет ли предоставлено разрешение.

#### 5.2.2.1 Роли RBAC

Семь стандартных ролей разделены на две большие категории в зависимости от того, может ли данная роль работать с элементами, которые считаются «чувствительными к безопасности». Ресурсы, атрибуты и операции, которые могут повлиять на административную безопасность (например, ресурсы области безопасности и атрибуты, содержащие пароли), являются «чувствительными к безопасности».

Четырем ролям не предоставлены разрешения для "чувствительных к безопасности" элементов:

– «Монитор» – роль, доступная только для чтения. Невозможно изменить какой-либо ресурс;

– «Разрешения оператора» – монитора, а также может изменять состояние среды выполнения, но не может изменять ничего, что в конечном итоге попадает в постоянную конфигурацию. Можно, например, перезапустить сервер;

– «Разрешения сопровождающего» – оператора, а также возможность изменять постоянную конфигурацию;

– «Развертыватель» – как и сопровождающий, но с разрешением изменять постоянную конфигурацию, ограниченным ресурсами, которые считаются «ресурсами приложения».

«Развертывание» - это ресурс приложения. Сервер обмена сообщениями таковым не является. Такие элементы, как источники данных и пункты назначения обмена сообщениями в Джакарте, по умолчанию не считаются ресурсами приложения, но это можно настроить.

Трем ролям предоставляются разрешения для доступа к элементам, чувствительным к безопасности:

– «Суперпользователь» – имеет все права доступа;

– «Администратор» – имеет все разрешения, за исключением того, что не может читать или записывать ресурсы, относящиеся к системе ведения журнала административного аудита;

– «Аудитор» – может читать все, что угодно. Может изменять только ресурсы, относящиеся к системе ведения журнала административного аудита.

Роли аудитора и администратора предназначены для организаций, которым требуется разделение обязанностей между теми, кто проверяет обычные административные действия, и теми, кто их выполняет, при этом те, кто выполняет большинство действий (роль администратора), не могут читать или изменять конфигурацию аудита.

#### 5.2.2.2 Ограничения контроля доступа

Следующие факторы используются для определения того, предоставлено ли данной роли разрешение:

– что представляет собой запрашиваемое действие (адрес, чтение, запись);

– влияет ли ресурс, атрибут или операция на постоянную конфигурацию;

– независимо от того, связан ли ресурс, атрибут или операция с функцией ведения журнала административного аудита;

– независимо от того, настроен ли ресурс, атрибут или операция как чувствительные к безопасности;

– содержит ли значение атрибута или операционного параметра выражение хранилища безопасности;

– считается ли ресурс связанным с приложениями, а не являющимся частью общей конфигурации контейнера.

Первые три из перечисленных факторов не поддаются настройке; последние три - допускают некоторую настройку. Подробности см. в разделе «Настройка ограничений».

#### 5.2.2.3 Обращение к ресурсу

Как упоминалось выше, разрешения предоставляются для выполнения одного из трех действий: обращения к ресурсу, его чтения и изменения. Последние два действия не требуют пояснений. Но что подразумевается под «обращением» к ресурсу?

«Обращение» к ресурсу означает выполнение действия, которое позволяет пользователю определить существует ли ресурс по данному адресу на самом деле. Например, операция «чтение имен дочерних элементов» позволяет пользователю определить действительные адреса. Попытка прочитать ресурс и получение сообщения об ошибке «В разрешении отказано» также дает пользователю ключ к пониманию того, что по запрашиваемому адресу действительно находится ресурс.

Некоторые ресурсы могут содержать конфиденциальную информацию в качестве части своего адреса. Например, ресурсы области безопасности включают имя области в качестве последнего элемента в адресе. Это имя области потенциально чувствительно к безопасности; например, оно является частью данных, используемых при создании хэша пароля пользователя. Поскольку некоторые адреса могут содержать конфиденциальные данные, пользователю необходимо разрешение даже для «обращения» к ресурсу. Если пользователь попытается обратиться к ресурсу и у него не будет разрешения, он не получит сообщение об ошибке типа «отказано в разрешении». Скорее всего, система будет реагировать так, как будто ресурс даже не существует, например, исключая ресурс из результата операции «чтение

дочерних имен» или выдавая ошибку «Такого ресурса нет» вместо «Отказано в разрешении», если пользователь пытается прочитать или записать ресурс.

Другим термином, обозначающим «обращение» к ресурсу, является «поиск» этого ресурса.

### 5.2.3 Переключение на провайдера «rbac»

Используйте интерфейс командной строки для переключения поставщика управления доступом.

**Внимание:** перед сменой поставщика «ролей», убедитесь, что ваши настройки пользователя, будет отображаться на одну из ролей RBAC, предпочтительно с по меньшей мере одним в Роль администратора или суперпользователя. В противном случае вашей установкой можно будет управлять только завершив ее и отредактировав конфигурацию «xml». Если вы запустили одну из стандартных конфигураций «xml», поставляемых с WildBoss Pro, пользователь «$local» будет сопоставлен с ролью «Суперпользователь» и будет включена схема аутентификации «локальная». Это позволит пользователю, использующему CLI в той же системе, что и процесс WildBoss Pro, получить полные административные разрешения. Удаленные пользователи CLI и пользователи веб-консоли администратора не будут иметь никаких разрешений.

Мы рекомендуем сопоставить хотя бы одного пользователя, кроме «$local», перед переключением провайдера на «rbac». Вы можете выполнить все настройки, связанные с провайдером «rbac», даже если для него установлено значение «simple».

Ресурсы управления, связанные с контролем доступа, расположены в разделе «coreservice= management/access=authorization» дерева ресурсов управления. Обновите атрибут «provider», чтобы выбрать поставщика «simple» и «rbac». Для вступления в силу любого обновления требуется перезагрузка или перезапуск компьютера.

```
[standalone@localhost:9990 /] cd core-
service=management/access=authorization
[standalone@localhost:9990 access=authorization] :writeattribute(
name=provider,value=rbac)
{
 "outcome" => "success",
 "response-headers" => {
   "operation-requires-reload" => true,
   "process-state" => "reload-required"
 }
}
[standalone@localhost:9990 access=authorization] reload
```

В управляемом домене конфигурация контроля доступа является частью конфигурации всего домена, поэтому адрес ресурса такой же, как указано выше, но интерфейс командной строки подключен к главному домену «Контроллер»:

```
[domain@localhost:9990 /] cd core-
service=management/access=authorization
[domain@localhost:9990 access=authorization] :writeattribute(
name=provider,value=rbac)
{
 "outcome" => "success",
 "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
 },
 "result" => undefined,
```

```
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
     "server-one" => {"response" => {
     "outcome" => "success",
        "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
        }
     }},
     "server-two" => {"response" => {
     "outcome" => "success",
        "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
        }
     }}
   }}}}
}
[domain@localhost:9990 access=authorization] reload --host=master
```

Как и в случае с автономным сервером, для вступления изменений в силу требуется перезагрузка или перезапуск. В этом случае потребуется перезагрузить или перезапустить все хосты и серверы в домене, начиная с главного Контроллер домена, поэтому обязательно тщательно спланируйте, прежде чем вносить это изменение.

### 5.2.4 Сопоставление пользователей и групп с ролями

Как только поставщик управления доступом «rbac» будет включен, только пользователи, которым назначена одна из доступных ролей, будут иметь какие-либо административные разрешения. Таким образом, чтобы RBAC был полезен, необходимо выполнить сопоставление между отдельными пользователями или группами пользователей и доступными ролями.

#### 5.2.4.1 Сопоставление отдельных пользователей

Самый простой способ привязать отдельных пользователей к ролям - это использовать веб-консоль администратора.

Перейдите на вкладку «Администрирование» и вложенную вкладку «Пользователи». Оттуда можно добавлять, удалять или редактировать привязки отдельных пользователей.

[images/usermapping.png]

Интерфейс командной строки также может использоваться для сопоставления отдельных пользователей с ролями.

Сначала, если таковая не существует, создайте родительский ресурс для всех сопоставлений для роли. Здесь мы создаем ресурс для роли администратора.

```
[domain@localhost:9990 /] /core-
service=management/access=authorization/rolemapping=
Administrator:add
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}}
}
```

Как только это будет сделано, сопоставьте пользователя с этой ролью:

```
[domain@localhost:9990 /] /core-
service=management/access=authorization/rolemapping=
Administrator/include=user-jsmith:add(name=jsmith,type=USER)
{
 "outcome" => "success",
 "result" => undefined,
 "server-groups" => {"main-server-group" => {"host" => {"master" => {
   "server-one" => {"response" => {"outcome" => "success"}},
   "server-two" => {"response" => {"outcome" => "success"}}
 }}}}
}
```

Теперь, если пользователь «jsmith» пройдет аутентификацию в любом домене безопасности, связанном с используемым им интерфейсом управления, ему будет присвоена роль администратора.

#### 5.2.4.2 Группы пользователей

"Группа" - это произвольный набор пользователей, которые могут существовать в среде конечного пользователя. Они могут называться так, как того хочет организация конечного пользователя, и могут содержать любых пользователей, которых хочет организация конечного пользователя. Некоторые типы хранилищ данных для проверки подлинности, поддерживаемые WildBoss Pro security realms, позволяют получать доступ к информации о том, к каким группам принадлежит пользователь, и связывать эту информацию с объектом, созданным при проверке подлинности пользователя. В настоящее время это поддерживается для следующих типов хранилищ данных для проверки подлинности:

– файл свойств (через файл «<realm_name>-groups.properties»)

– LDAP (через конфигурацию, зависящую от каталога и сервера).

Группы удобны, когда дело доходит до привязки пользователя к роли, поскольку целые группы могут быть связаны с ролью в одном сопоставлении.

#### 5.4.2.3 Сопоставление групп с ролями

Самый простой способ сопоставить группы с ролями - это использовать веб-консоль администратора.

Перейдите на вкладку «Администрирование» и во вложенную вкладку «Группы». Оттуда можно добавлять, удалять или редактировать сопоставления групп.

[images/groupmapping.png]

Интерфейс командной строки также может использоваться для сопоставления групп с ролями. Единственное отличие от сопоставления отдельных пользователей заключается в том, что значение атрибута «type» должно быть GROUP, а не USER.

```
[domain@localhost:9990 /] /core-
service=management/access=authorization/rolemapping=
Administrator/include=group-
SeniorAdmins:add(name=SeniorAdmins,type=GROUP)
{
 "outcome" => "success",
 "result" => undefined,
 "server-groups" => {"main-server-group" => {"host" => {"master" => {
   "server-one" => {"response" => {"outcome" => "success"}},
   "server-two" => {"response" => {"outcome" => "success"}}
 }}}}
}
```

#### 5.4.2.4 Включение всех прошедших проверку подлинности пользователей в роль

Можно указать, что все прошедшие проверку подлинности пользователи должны быть привязаны к определенной роли. Это может быть использовано, например, для обеспечения того, чтобы каждый, кто может пройти проверку подлинности, мог, по крайней мере, иметь права мониторинга.

**Внимание:** пользователь, который может пройти аутентификацию в области безопасности управления, но не привязан к какой-либо роли, не сможет выполнять никаких административных функций, даже читать.

В веб-консоли администратора перейдите на вкладку «Администрирование», вложенную вкладку «Роли», выделите соответствующую роль, нажмите кнопку «Редактировать» и установите флажок «Включить все»:

[images/includeall.png]

То же самое изменение можно внести с помощью интерфейса командной строки:

```
[domain@localhost:9990 /] /core-
service=management/access=authorization/rolemapping=
Monitor:write-attribute(name=include-all,value=true)
{
 "outcome" => "success",
 "result" => undefined,
 "server-groups" => {"main-server-group" => {"host" => {"master" => {
  "server-one" => {"response" => {"outcome" => "success"}},
  "server-two" => {"response" => {"outcome" => "success"}}
 }}}}
}
```

#### 5.4.2.5 Исключение пользователей и групп

Также можно явно исключить определенных пользователей и группы из роли. Исключения имеют приоритет над включениями, включая случаи, когда для атрибута «include-all» установлено значение «true» для роли.

В консоли администратора исключения выполняются на тех же экранах, что и включения. В диалоговом окне добавления просто измените выпадающий список «Тип» на «Исключить».

[images/excludemapping.png]

В интерфейсе командной строки «excludes» идентичны «includes», за исключением того, что адрес ресурса имеет «exclude» вместо «include» в качестве ключа для последнего элемента «address»:

```
[domain@localhost:9990 /] /core-
service=management/access=authorization/rolemapping=
Monitor/exclude=group-Temps:add(name=Temps,type=GROUP)
{
 "outcome" => "success",
 "result" => undefined,
 "server-groups" => {"main-server-group" => {"host" => {"master" => {
   "server-one" => {"response" => {"outcome" => "success"}},
   "server-two" => {"response" => {"outcome" => "success"}}
 }}}}
}
```

#### 5.4.2.6 Пользователи, которые соответствуют нескольким ролям

Возможно, что данному пользователю будет присвоено более одной роли. В этом случае по умолчанию пользователю будет предоставлено объединение разрешений для двух

ролей. Это поведение может быть изменено на глобальной основе, чтобы вместо этого отвечать на запрос пользователя сообщением об ошибке, если будет обнаружена такая ситуация:

```
[standalone@localhost:9990 /] cd core-
service=management/access=authorization
[standalone@localhost:9990 access=authorization] :write-
attribute(name=permissioncombination-
policy,value=rejecting)
{"outcome" => "success"}
```

Обратите внимание, что перезагрузка не требуется; изменение вступает в силу немедленно.

Чтобы восстановить поведение по умолчанию, установите значение «permissive»:

```
[standalone@localhost:9990 /] cd core-
service=management/access=authorization
[standalone@localhost:9990 access=authorization] :write-
attribute(name=permissioncombination-
policy,value=permissive)
{"outcome" => "success"}
```

### 5.2.5 Добавление пользовательских ролей в управляемом домене

Управляемый домен может включать в себя множество серверов с различными конфигурациями и размещением различных приложений. В такой среде, вероятно, будут разные команды администраторов, ответственных за разные части домена. Чтобы позволить организациям предоставлять разрешения только для части домена, схема RBAC WildBoss Pro позволяет создавать пользовательские «ограниченные роли». Ограниченные роли основаны на семи стандартных ролях, но с разрешениями, ограниченными частью домена – либо набором групп серверов, либо набором хостов.

#### 5.2.5.1 Роли, ограниченные областью действия группы серверов

Привилегии для роли, относящейся к группе серверов, ограничены ресурсами, связанными с одной или несколькими группами серверов. Группы серверов часто связаны с определенным приложением или набором приложений; организации, в которых есть отдельные команды, ответственные за разные приложения, могут счесть полезными роли, относящиеся к группе серверов.

Роль, относящаяся к группе серверов, эквивалентна роли по умолчанию, на которой она основана, но с привилегиями, ограниченными для целевых ресурсов в деревьях ресурсов, корнями уходящих в ресурсы группы серверов. Роль, относящаяся к группе серверов, может быть настроена таким образом, чтобы включать привилегии для следующих деревьев ресурсов, логически связанных с группой серверов:

– «Профиль»;
– «Группа привязки сокетов»;
– «Развертывание»
– «Переопределение развертывания»;
– «Группа серверов»;
– «Конфигурация сервера»;
– «Сервер».

Ресурсы в профиле, группе привязки сокетов, конфигурации сервера и серверных частях дерева, которые логически не связаны с группой серверов, связанной с ролью в области

действия группы серверов, не будут доступны пользователю в этой роли. Таким образом, в домене с группами серверов «a» и «b» пользователь с ролью в области «servergroup», которая предоставляет доступ к «a», не сможет обратиться к «/server-group=b». Система будет рассматривать этот ресурс как несуществующий для данного пользователя.

В дополнение к этим привилегиям пользователи с ролью, относящейся к группе серверов, будут иметь привилегии на чтение конфиденциальных данных (эквивалентные роли мониторинга) для ресурсов, отличных от перечисленных выше.

Самый простой способ создать роль, относящуюся к группе серверов - это использовать консоль администратора. Но вы также можете использовать интерфейс командной строки для создания роли, относящейся к группе серверов.

```
[domain@localhost:9990 /] /core-
service=management/access=authorization/server-groupscoped-
role=MainGroupAdmins:add(base-role=Administrator,server-groups=[main-
servergroup])
{
 "outcome" => "success",
 "result" => undefined,
 "server-groups" => {"main-server-group" => {"host" => {"master" => {
   "server-one" => {"response" => {"outcome" => "success"}},
   "server-two" => {"response" => {"outcome" => "success"}}
 }}}}
}
```

Как только роль создана, пользователи или группы могут быть привязаны к ней так же, как и к семи стандартным ролям.

#### 5.2.5.2 Роли, ограниченные областью действия хоста

Привилегии для роли в области хоста ограничены ресурсами, связанными с одним или несколькими хостами. Пользователь с ролью в области хоста не может изменять конфигурацию всего домена. Организации могут использовать роли, привязанные к узлу, чтобы предоставить администраторам относительно широкие административные права для узла, не предоставляя таких прав всему управляемому домену.

Роль в области хоста эквивалентна роли по умолчанию, на которой она основана, но с привилегиями, ограниченными для целевых ресурсов в деревьях ресурсов, связанных с ресурсами хоста, для одного или нескольких указанных хостов.

В дополнение к этим привилегиям пользователи с ролью, ограниченной областью действия хоста, будут иметь конфиденциальные права на чтение (эквивалентно роли монитора) для ресурсов всего домена (т.е. тех, которые не находятся в разделе «/host=*» дерева).

Ресурсы в части дерева «/host=*», которые не связаны с хостами, указанными для данного хоста Роль с ограниченной областью действия не будет видна пользователям с этой ролью, относящейся к области действия хоста. Таким образом, в домене с хостами «a» и «b» пользователь с ролью в области хоста, предоставляющий доступ к «a», не сможет обратиться к «/host=b». Система будет рассматривать этот ресурс как несуществующий для этого пользователя.

Самый простой способ создать роль с областью действия хоста - это использовать консоль администратора. Но вы также можете использовать интерфейс командной строки для создания роли с областью действия хоста.

```
[domain@localhost:9990 /] /core-
service=management/access=authorization/host-scopedrole=
MasterOperators:add(base-role=Operator,hosts=[master]}
```

```
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}}
}
```

Как только роль создана, пользователи или группы могут быть привязаны к ней так же, как и к семи стандартным ролям.

#### 5.2.5.3 Использование консоли администратора для создания ограниченных ролей

Роли, относящиеся как к группе серверов, так и к хосту, можно добавлять, удалять или редактировать с помощью консоли администратора. Выберите «Роли, относящиеся к области действия» на вкладке «Администрирование», вложенной вкладке «Роли»:

[images/scopedroles.png]

При добавлении новой ограниченной роли используйте выпадающий список «Тип» в диалоговом окне, чтобы выбрать между ролью, относящейся к области хоста, и ролью, относящейся к группе серверов. Затем поместите имена соответствующих хостов или групп серверов в текстовое поле «Область».

[images/addscopedrole.png]

### 5.2.6 Настройка ограничений

Следующие факторы используются для определения того, предоставлено ли данной роли разрешение:

– что такое запрашиваемое действие (адрес, чтение, запись);

– влияет ли ресурс, атрибут или операция на постоянную конфигурацию;

– независимо от того, связан ли ресурс, атрибут или операция с функцией ведения журнала административного аудита;

– независимо от того, настроен ли ресурс, атрибут или операция как чувствительные к безопасности;

– содержит ли значение атрибута или операционного параметра выражение хранилища безопасности или зашифрованное выражение;

– считается ли ресурс связанным с приложениями, а не являющимся частью общей конфигурации контейнера.

Первые три из этих факторов не настраиваются; последние три допускают некоторую настройку.

#### 5.2.6.1 Настройка чувствительности

Ограничения «чувствительности» связаны с ограничением доступа к конфиденциальным данным. В разных организациях могут быть разные мнения о том, что является конфиденциальным, поэтому WildBoss Pro предоставляет параметры конфигурации, позволяющие пользователям адаптировать эти ограничения.

***Важные ресурсы, атрибуты и операции.***

Разработчики ядра WildBoss Pro и любой другой подсистемы могут добавлять к ресурсам, атрибутам или операциям «классификацию чувствительности». Классификации либо предоставляются ядром и могут применяться в любом месте модели управления, либо относятся к определенной подсистеме. Для каждой классификации будет указан параметр, определяющий, считаются ли по умолчанию действия с адресацией, чтением и записью конфиденциальными. Если действие является конфиденциальным, разрешения будут иметь

только пользователи с ролями, которые могут работать с конфиденциальными данными (администратор, аудитор, суперпользователь).

Используя интерфейс командной строки, администраторы могут просматривать настройки для классификации. Например, существует базовая классификация под названием «socket-config», которая применяется ко всем элементам модели, относящимся к настройке сокетов:

```
[domain@localhost:9990 /] cd coreservice=
management/access=authorization/constraint=sensitivityclassification/
type=core/classification=socket-config
[domain@localhost:9990 classification=socket-config] ls -l
ATTRIBUTE VALUE TYPE
configured-requires-addressable undefined BOOLEAN
configured-requires-read undefined BOOLEAN
configured-requires-write undefined BOOLEAN
default-requires-addressable false BOOLEAN
default-requires-read false BOOLEAN
default-requires-write true BOOLEAN
CHILD MIN-OCCURS MAX-OCCURS
applies-to n/a n/a
```

Различные атрибуты «default-requires-…» указывают, должен ли пользователь выполнять роль, разрешающую действия, связанные с безопасностью, для выполнения действия. В приведенном выше примере конфигурации сокета значение «default-requires-write» равно «true», в то время как другие значения – «false». Таким образом, по умолчанию изменение параметра, связанного с конфигурацией сокета, считается конфиденциальным, в то время как обращение к этим ресурсам или выполнение операций чтения не являются конфиденциальными.

Атрибуты, требуемые по умолчанию, доступны только для чтения. Однако настроенные атрибуты, требуемые по умолчанию, можно изменить, чтобы заменить настройки по умолчанию на те, которые подходят для вашей организации. Например, если ваша организация не считает изменение параметров конфигурации сокета важным с точки зрения безопасности, вы можете изменить этот параметр:

```
[domain@localhost:9990 classification=socket-config] :write-
attribute(name=configuredrequires-
write,value=false)
{
 "outcome" => "success",
 "result" => undefined,
 "server-groups" => {"main-server-group" => {"host" => {"master" => {
  "server-one" => {"response" => {"outcome" => "success"}},
  "server-two" => {"response" => {"outcome" => "success"}}
 }}}}
}
```

Администраторы также могут ознакомиться с моделью управления, чтобы увидеть, к каким ресурсам, атрибутам и операциям применяется та или иная классификация уязвимостей:

```
[domain@localhost:9990 classification=socket-config] :read-children-
resources(childtype=
applies-to)
{
 "outcome" => "success",
 "result" => {
  "/host=master" => {
```

```
    "address" => "/host=master",
    "attributes" => [],
    "entire-resource" => false,
    "operations" => ["resolve-internet-address"]
  },
  "/host=master/core-service=host-environment" => {
    "address" => "/host=master/core-service=host-environment",
    "attributes" => [
      "host-controller-port",
      "host-controller-address",
      "process-controller-port",
      "process-controller-address"
    ],
    "entire-resource" => false,
    "operations" => []
  },
  "/host=master/core-service=management/management-
  interface=http-interface" =>
  {
    "address" => "/host=master/core-
    service=management/managementinterface=
    http-interface",
    "attributes" => [
    "port",
    "secure-interface",
    "secure-port",
    "interface"
    ],
    "entire-resource" => false,
    "operations" => []
  },
  "/host=master/core-service=management/management-
  interface=native-interface"
  => {
    "address" => "/host=master/core-
    service=management/managementinterface=
    native-interface",
    "attributes" => [
    "port",
    "interface"
    ],
    "entire-resource" => false,
    "operations" => []
  },
  "/host=master/interface=*" => {
    "address" => "/host=master/interface=*",
    "attributes" => [],
    "entire-resource" => true,
    "operations" => ["resolve-internet-address"]
 },
 "/host=master/server-config=*/interface=*" => {
  "address" => "/host=master/server-config=*/interface=*",
  "attributes" => [],
  "entire-resource" => true,
```

```
   "operations" => []
  },
  "/interface=*" => {
   "address" => "/interface=*",
   "attributes" => [],
   "entire-resource" => true,
   "operations" => []
  },
  "/profile=*/subsystem=messaging/hornetq-server=*/broadcast-
  group=*" => {
    "address" => "/profile=*/subsystem=messaging/hornetq-
    server=*/broadcastgroup=*",
    "attributes" => [
    "group-address",
    "group-port",
    "local-bind-address",
    "local-bind-port"
    ],
    "entire-resource" => false,
    "operations" => []
  },
  "/profile=*/subsystem=messaging/hornetq-server=*/discovery-
  group=*" => {
    "address" => "/profile=*/subsystem=messaging/hornetq-
    server=*/discoverygroup=*",
    "attributes" => [
    "group-address",
    "group-port",
    "local-bind-address"
    ],
    "entire-resource" => false,
    "operations" => []
  },
  "/profile=*/subsystem=transactions" => {
    "address" => "/profile=*/subsystem=transactions",
    "attributes" => ["process-id-socket-max-ports"],
    "entire-resource" => false,
    "operations" => []
  },
  "/server-group=*" => {
    "address" => "/server-group=*",
    "attributes" => ["socket-binding-port-offset"],
    "entire-resource" => false,
    "operations" => []
  },
  "/socket-binding-group=*" => {
    "address" => "/socket-binding-group=*",
    "attributes" => [],
    "entire-resource" => true,
    "operations" => []
  }
  }
}
```

Для каждого адреса, к которому применяется классификация, будет создан отдельный дочерний элемент. Атрибут «entireresource» будет иметь значение «true», если классификация применяется ко всему ресурсу. В противном случае атрибуты «attributes» и «operations» будут содержать названия атрибутов или операций, к которым применяется классификация.

#### 5.2.6.2 Широко используемые классификации

Несколько основных классификаций чувствительности широко используются в рамках всей модели управления и заслуживают особого упоминания (таблица 1).

Таблица 1 – Классификация чувствительности

| Наименование | Описание |
|---|---|
| учетные данные | атрибут, значение которого является своего рода учетными данными, например паролем или именем пользователя. По умолчанию доступен как для чтения, так и для записи |
| безопасность-домен-ссылка | атрибут, значением которого является имя домена безопасности. По умолчанию доступен как для чтения, так и для записи |
| безопасность-область-ссылка | атрибут, значением которого является имя области безопасности. По умолчанию доступен как для чтения, так и для записи |
| привязка к сокету-ссылка | атрибут, значением которого является имя привязки к сокету. По умолчанию не чувствителен ни к какому действию |
| конфигурация сокета | ресурс, атрибут или операция, которые каким-либо образом связаны с настройкой сокета. По умолчанию уязвимы для записи |

***Значения с выражениями хранилища безопасности.***

По умолчанию любой атрибут или операционный параметр, значение которого включает выражение «security vault», будет рассматриваться как конфиденциальный, даже если классификация конфиденциальности не применяется или классификация не рассматривает действие как конфиденциальное.

Эту настройку можно глобально изменить с помощью интерфейса командной строки. Для этой конфигурации существует ресурс:

```
[domain@localhost:9990 /] cd coreservice=
management/access=authorization/constraint=vault-expression
[domain@localhost:9990 constraint=vault-expression] ls -l
ATTRIBUTE VALUE TYPE
configured-requires-read undefined BOOLEAN
configured-requires-write undefined BOOLEAN
default-requires-read true BOOLEAN
default-requires-write true BOOLEAN
```

Различные атрибуты «default-requires-...» указывают, должен ли пользователь выполнять роль, разрешающую действия, связанные с безопасностью, для выполнения действия. Таким образом, по умолчанию для атрибутов чтения и записи, значения которых

включают выражения хранилища, требуется, чтобы пользователь выполнял одну из ролей с разрешениями на конфиденциальные данные.

Атрибуты, требуемые по умолчанию, доступны только для чтения. Однако настроенные атрибуты, требуемые по умолчанию, могут быть изменены, чтобы заменить параметры по умолчанию параметрами, подходящими для вашей организации. Например, если ваша организация не считает чтение выражений хранилища важным фактором безопасности, вы можете изменить этот параметр:

```
[domain@localhost:9990 constraint=vault-expression] :write-
attribute(name=configuredrequires-
read,value=false)
{
 "outcome" => "success",
 "result" => undefined,
 "server-groups" => {"main-server-group" => {"host" => {"master" => {
 "server-one" => {"response" => {"outcome" => "success"}},
 "server-two" => {"response" => {"outcome" => "success"}}
 }}}}
}
```

**Внимание:** это ограничение на выражение хранилища в некоторой степени совпадает с основной классификацией чувствительности «учетных данных» в том смысле, что наиболее типичное использование выражения хранилища - это атрибуты, содержащие имя пользователя или пароль, и они, как правило, будут помечены классификацией чувствительности «учетных данных». Таким образом, если вы измените настройки для классификации чувствительности «учетных данных», вам также может потребоваться внести соответствующие изменения в настройки ограничения выражения хранилища, иначе ваши изменения не будут иметь полного эффекта.

Однако имейте в виду, что выражения «vault» могут использоваться в любом атрибуте, который поддерживает выражения, а не только в атрибутах типа учетных данных. Поэтому важно ознакомиться с тем, где и как ваша организация использует выражения «vault», прежде чем изменять эти настройки.

#### 5.2.6.3 Настройка доступа к роли "Развертывателя"

Права на запись для стандартной роли «Развертывателя» ограничены ресурсами, которые считаются «ресурсами приложения», т.е. концептуально являются частью приложения, а не частью общей конфигурации сервера. По умолчанию только ресурсы развертывания считаются ресурсами приложения. Однако у разных организаций могут быть разные мнения о том, что считается ресурсом приложения, поэтому для типов ресурсов, которые авторы подсистем считают потенциально ресурсами приложения, WildBoss Pro предоставляет опцию конфигурации, позволяющую объявить их таковыми. Такие ресурсы будут помечены как «классификация приложений».

Например, почтовая подсистема предоставляет такую классификацию:

```
[domain@localhost:9990 /] cd /coreservice=
management/access=authorization/constraint=applicationclassification/
type=mail/classification=mail-session
[domain@localhost:9990 classification=mail-session] ls -l
ATTRIBUTE VALUE TYPE
configured-application undefined BOOLEAN
default-application false BOOLEAN
CHILD MIN-OCCURS MAX-OCCURS
applies-to n/a n/a
```

Используйте «read-resource» или «read-children-resources», чтобы увидеть, к каким ресурсам применяется эта классификация:

```
[domain@localhost:9990 classification=mail-session] :read-children-
resources(childtype=
applies-to)
{
 "outcome" => "success",
 "result" => {"/profile=*/subsystem=mail/mail-session=*" => {
 "address" => "/profile=*/subsystem=mail/mail-session=*",
 "attributes" => [],
 "entire-resource" => true,
 "operations" => []
}}
}
```

Это указывает на то, что данная классификация, как представляется, применима только к ресурсам почтовой подсистемы «mailsession».

Чтобы пользователи в роли разработчика могли записывать ресурсы с этой классификацией, установите для атрибута «configured-application» значение «true».

```
[domain@localhost:9990 classification=mail-session] :write-
attribute(name=configuredapplication,
value=true)
{
 "outcome" => "success",
 "result" => undefined,
 "server-groups" => {"main-server-group" => {"host" => {"master" => {
 "server-one" => {"response" => {"outcome" => "success"}},
 "server-two" => {"response" => {"outcome" => "success"}}
}}}}
}
```

Классификация приложений, поставляемых вместе с WildBoss Pro.

Подсистемы, поставляемые с полным дистрибутивом WildBoss Pro, включают в себя следующие категории приложений (Таблица 2):

Таблица 2 – Категории приложений

| Подсистема | Классификация |
|---|---|
| источники данных | источник данных |
| источники данных | источник данных |
| источники данных | источник данных |
| регистрация | регистратор |
| регистрация | ведение журнала-профиль |
| почта | почтовый сеанс |
| обмен сообщениями | jms-очередь |
| обмен сообщениями | jms-тема |
| обмен сообщениями | очередь |
| обмен сообщениями | настройка безопасности |
| присвоение имен | связующий |
| адаптеры ресурсов | ресурс-адаптер |
| безопасность | безопасность-домен |

В каждом случае классификация применяется к ресурсам, которые вы ожидаете получить, учитывая ее название.

### 5.2.7 Влияние RBAC на работу администратора с пользователями

Схема RBAC приведет к уменьшению разрешений для администраторов, которые не соответствуют роли суперпользователя, поэтому это, конечно, окажет некоторое влияние на их опыт использования административных инструментов, таких как консоль администратора и интерфейс командной строки.

#### 5.2.7.1 Консоль администратора

Консоль администратора прилагает все усилия, чтобы обеспечить удобный пользовательский интерфейс, даже если у пользователя ограниченные права доступа. Ресурсы, доступ к которым пользователю запрещен, просто не будут отображаться или, при необходимости, будут заменены в пользовательском интерфейсе указанием на то, что пользователь не авторизован. Элементы взаимодействия, такие как кнопки «Добавить» и «Удалить», а также ссылки «Редактировать», будут недоступны, если у пользователя нет прав на запись.

#### 5.2.7.2 CLI

Интерфейс командной строки - это гораздо более свободный инструмент, чем консоль администратора, позволяющий пользователям пытаться выполнять любые операции, которые они пожелают, поэтому более вероятно, что пользователи, которые попытаются выполнить действия, для которых у них нет необходимых разрешений, получат сообщения об ошибках. Например, пользователь в роли «Наблюдателя» не может читать пароли:

```
[domain@localhost:9990 /]
/profile=default/subsystem=datasources/datasource=
ExampleDS:read-attribute(name=password)
{
 "outcome" => "failed",
 "result" => undefined,
 "failure-description" => "WFLYCTL0313: Unauthorized to execute
 operation 'readattribute'
 for resource '[
 (\"profile\" => \"default\"),
 (\"subsystem\" => \"datasources\"),
 (\"data-source\" => \"ExampleDS\")
 ]' -- \"WFLYCTL0332: Permission denied\"",
 "rolled-back" => true
}
```

Если пользователю даже не разрешено обращаться к ресурсу, то ответ будет таким, как будто ресурс не существует, хотя на самом деле он существует:

```
[domain@localhost:9990 /]
/profile=default/subsystem=security/securitydomain=
other:read-resource
{
 "outcome" => "failed",
 "failure-description" => "WFLYCTL0216: Management resource '[
 (\"profile\" => \"default\"),
 (\"subsystem\" => \"security\"),
 (\"security-domain\" => \"other\")
 ]' not found",
```

```
  "rolled-back" => true
}
```

Это предотвращает поиск конфиденциальных данных в адресах ресурсов неавторизованными пользователями, проверяя их на наличие ошибок типа «Отказано в разрешении».

Пользователи, использующие операцию чтения ресурсов, могут запрашивать данные, некоторые из которых им разрешено просматривать, а некоторые - нет. Если это произойдет, запрос не завершится ошибкой, но недоступные данные будут удалены, а в заголовок ответа будет включена информация о том, что не было включено. Здесь мы показываем, как монитор пытается рекурсивно считывать конфигурацию подсистемы безопасности:

```
[domain@localhost:9990 /]
/profile=default/subsystem=security:readresource(
recursive=true)
{
  "outcome" => "success",
  "result" => {
    "deep-copy-subject-mode" => undefined,
    "security-domain" => undefined,
    "vault" => undefined
  },
    "response-headers" => {"access-control" => [{
    "absolute-address" => [
    ("profile" => "default"),
    ("subsystem" => "security")
    ],
    "relative-address" => [],
    "filtered-attributes" => ["deep-copy-subject-mode"],
    "filtered-children-types" => ["security-domain"]
  }]}
}
```

Раздел заголовков ответов содержит данные управления доступом в виде списка, содержащего по одному элементу для каждого соответствующего ресурса. (В данном случае только один). Отображается абсолютный и относительный адрес ресурса, а также тот факт, что значение атрибута «deep-copy-subject-mode» было отфильтровано (т.е. в качестве значения отображается значение «undefined», которое может не соответствовать реальному значению), а также тот факт, что дочерние ресурсы типа «security-domain» был отфильтрован.

#### 5.2.7.3 Описание ограничений контроля доступа в модели управления

***Метаданные.***

Описательные метаданные модели управления, возвращаемые в результате таких операций, как «read-resourcedescription» и «read-operation-description», могут быть сконфигурированы таким образом, чтобы включать информацию, описывающую ограничения контроля доступа, относящиеся к ресурсу, это делается с помощью параметра «access-control». Выходные данные будут адаптированы к разрешениям вызывающего абонента. Например, пользователь, которому назначена роль наблюдателя, может запросить информацию о ресурсе в почтовой подсистеме:

```
[domain@localhost:9990 /] cd
/profile=default/subsystem=mail/mailsession=
default/server=smtp
```

```
[domain@localhost:9990 server=smtp] :read-resource-description(access-
control=trimdescriptions)
{
   "outcome" => "success",
   "result" => {
   "description" => undefined,
     "access-constraints" => {"application" => {"mail-session" =>
     {"type" =>
     "mail"}}},
     "attributes" => undefined,
     "operations" => undefined,
     "children" => {},
     "access-control" => {
       "default" => {
        "read" => true,
        "write" => false,
        "attributes" => {
          "outbound-socket-binding-ref" => {
            "read" => true,
            "write" => false
          },
          "username" => {
            "read" => false,
            "write" => false
          },
          "tls" => {
            "read" => true,
            "write" => false
          },
          "ssl" => {
            "read" => true,
            "write" => false
          },
          "password" => {
            "read" => false,
            "write" => false
          }
     }
    },
    "exceptions" => {}
   }
  }
}
```

Поскольку в качестве значения параметра «access-control» использовалось значение «trim-descriptions», типичные данные «описание», «атрибуты», «операции» и «дочерние элементы» в основном не используются. (Подробнее об этом смотрите ниже). Поле «Ограничения доступа» указывает на то, что к этому ресурсу добавлено ограничение приложения. Поле «Управление доступом» содержит информацию о разрешениях, которые текущий пользователь имеет для этого ресурса. В разделе «По умолчанию» приведены настройки по умолчанию для ресурсов этого типа. Поля «Чтение» и «запись» непосредственно в поле «По умолчанию» показывают, что вызывающий объект может, в общем, считывать этот ресурс, но не может записывать его. В разделе «Атрибуты» отображаются настройки отдельных атрибутов. Обратите внимание, что монитор не может считывать атрибуты имени пользователя и пароля.

Существует три допустимых значения для параметра управления доступом: «read-resource-description» и «read-operation-description»:

– **«none»** – не включать информацию об управлении доступом в ответ. Это поведение по умолчанию, если параметр не указан;

– **«trim-descriptions»** – удалить обычные детали описания, как показано в примере выше;

– **«combined-descriptions»** - включают как обычные выходные данные, так и данные контроля доступа.

### 5.2.8 Изучение ваших собственных ролевых схем

Пользователи могут узнать, в каких ролях они работают. В консоли администратора нажмите на свое имя в правом верхнем углу; будут показаны роли, в которых вы работаете.

[images/callersroles.png]

Пользователи CLI должны использовать операцию «whoami» с набором атрибутов «verbose»:

```
[domain@localhost:9990 /] :whoami(verbose=true)
{
 "outcome" => "success",
 "result" => {
   "identity" => {
     "username" => "aadams",
     "realm" => "ManagementRealm"
   },
   "mapped-roles" => [
   "Maintainer"
   ]
 }
}
```

### 5.2.9 Возможность "Запуска от имени" для суперпользователей

Если пользователь сопоставляется с ролью суперпользователя, WildBoss Pro также поддерживает возможность запроса этого пользователя на сопоставление с одной или несколькими другими ролями. Это может быть полезно при проведении демонстраций или когда суперпользователь изменяет конфигурацию RBAC и хочет посмотреть, какой эффект окажут изменения с точки зрения пользователя в другой роли. Эта возможность доступна только для роли суперпользователя, поэтому ее можно использовать только для ограничения прав пользователя, а не для их потенциального увеличения.

#### 5.2.9.1 Запуск CLI

В CLI возможность запуска от имени зависит от конкретного запроса. Это делается с помощью заголовка операции «роли», значением которого может быть имя отдельной роли или список имен ролей, заключенный в квадратные скобки и разделенный запятыми.

Пример с низкоуровневой операцией:

```
[standalone@localhost:9990 /]
:whoami(verbose=true){roles=[Operator,Auditor]}
{
 "outcome" => "success",
 "result" => {
   "identity" => {
   "username" => "$local",
```

```
    "realm" => "ManagementRealm"
    },
    "mapped-roles" => [
     "Auditor",
     "Operator"
    ]
  }
}
```

Пример с командой CLI:

```
[standalone@localhost:9990 /] deploy /tmp/helloworld.war --
headers={roles=Monitor}
{"WFLYCTL0062: Composite operation failed and was rolled back. Steps
that failed:" =>
{"Operation step-1" => "WFLYCTL0313: Unauthorized to execute operation
'add' for
resource '[(\"deployment\" => \"helloworld.war\")]' -- \"WFLYCTL0332:
Permission
denied\""}}
[standalone@localhost:9990 /] deploy /tmp/helloworld.war --
headers={roles=Maintainer}
```

Здесь мы показываем эффект переключения на роль, которой не предоставлены необходимые разрешения.

#### 5.2.9.2 Запуск консоли администратора от имени

Пользователи консоли администратора могут изменить роль, в которой они работают, нажав на свое имя в правом верхнем углу и перейдя по ссылке «Run as…».

[images/callersroles.png]

Затем выберите роль, в которой вы хотите работать:

[images/runasrole.png]

Для того чтобы изменения вступили в силу, консоль необходимо будет перезапустить.

#### 5.2.9.3 Использование ролей запуска от имени поставщика «simple» контроля доступа

Эта возможность «run-as» доступна, даже если используется «simple» поставщик управления доступом. При использовании «simple» поставщика с любым прошедшим проверку подлинности администратором обращаются так же, как если бы он был назначен суперпользователем при использовании поставщика «rbac».

Однако «simple» провайдер на самом деле понимает все параметры конфигурации провайдера «rbac», описанные выше, но использует их только в том случае, если для запроса используется функция «run-as». В противном случае роль суперпользователя обладает всеми разрешениями, поэтому подробная настройка не имеет значения.

Использование возможности запуска от имени «simple» поставщика может быть полезно, если администратор настраивает конфигурацию поставщика «rbac» перед переключением поставщика на «rbac», чтобы эта конфигурация вступила в силу. Затем администратор может запускать различные роли, чтобы увидеть эффект от запланированных настроек.

# 6 Развертывание приложения

## 6.1 Управляемый домен

В управляемом домене развертывания связаны с группой серверов (см. «Основные концепции управления»). В этом случае развертывание будет доступно для любого сервера в группе серверов. Компоненты домена и контроллера хоста управляют распределением двоичных файлов по сети.

### 6.1.1 Команды развертывания

Распространение двоичных файлов развертывания включает в себя два этапа: загрузку развертывания в хранилище, которое контроллер домена будет использовать для распространения его содержимого, а затем назначение развертывания одной или нескольким группам серверов.

Вы можете сделать это одним движением с помощью CLI:

```
[domain@localhost:9990 /] deploy ~/Desktop/test-application.war
Either --all-server-groups or --server-groups must be specified.
[domain@localhost:9990 /] deploy ~/Desktop/test-application.war --all-
server-groups
'test-application.war' deployed successfully.
```

Развертывание будет доступно контроллеру домена, назначенному группе серверов, и развернуто на всех запущенных серверах в этой группе:

```
[domain@localhost:9990 /] :read-children-names(child-type=deployment)
{
 "outcome" => "success",
 "result" => [
   "mysql-connector-java-5.1.15.jar",
   "test-application.war"
 ]
}
[domain@localhost:9990 /] /server-group=main-server-
group/deployment=testapplication.
war:read-resource(include-runtime)
{
 "outcome" => "success",
 "result" => {
   "enabled" => true,
   "name" => "test-application.war",
   "managed" => true,
   "runtime-name" => "test-application.war"
 }
}
```

Если вы хотите, чтобы развертывание было выполнено только на серверах в некоторых группах серверов, но не на всех, используйте параметр «--server-groups» вместо «-all-server-groups»:

```
[domain@localhost:9990 /] deploy ~/Desktop/test-application.war --
server-groups=main
-server-group,another-group
'test-application.war' deployed successfully.
```

Если у вас есть новая версия развертывания, которую вы хотите развернуть вместо существующей, используйте параметр «--force»:

```
[domain@localhost:9990 /] deploy ~/Desktop/test-application.war --all-
server-groups
--force
'test-application.war' deployed successfully.
```

Вы можете удалить двоичные файлы из групп серверов с помощью команды отменить развертывание:

```
[domain@localhost:9990 /] undeploy test-application.war --all-
relevant-server-groups
Successfully undeployed test-application.war.
[domain@localhost:9990 /] /server-group=main-server-group:read-
children-names(childtype=
deployment)
{
 "outcome" => "success",
 "result" => []
}
```

Если вы хотите отменить развертывание только для некоторых групп серверов, но не для других, используйте параметр – «server-groups» вместо «-all-relevant-server-groups».

Команда CLI «deploy» поддерживает ряд других параметров, которые могут управлять поведением. Используйте параметр «--help», чтобы узнать больше:

```
[domain@localhost:9990 /] deploy --help
[...]
```

**Внимание:** управление развертываниями через веб-интерфейс обеспечивает альтернативный, иногда более простой подход.

### 6.1.2 Отдельные управляемые развертывания

Управляемые и неуправляемые развертывания могут быть «разнесены», то есть размещены в файловой системе в виде структуры каталогов, структура которой соответствует распакованной версии архива. Разнесенное развертывание может быть удобным в администрировании, если ваши административные процессы включают вставку или замену файлов из базовой версии для создания версии, адаптированной для конкретного использования (например, скопируйте в базовое развертывание, а затем скопируйте в файл «jboss-web.xml», чтобы адаптировать развертывание для использования в WildBoss Pro). Разнесенные развертывания - это также удобно в некоторых сценариях разработки, поскольку вы можете заменить файлы статического содержимого (например, «*.html», «*.css») при развертывании и сразу же получить доступ к новому содержимому, не требуя повторного развертывания.

Поскольку содержимое неуправляемого развертывания находится в вашем непосредственном ведении, следующие операции имеют смысл только для управляемого развертывания.

```
[domain@localhost:9990 /]
/deployment=exploded.war:add(content=[{empty=true}])
```

В результате будет создано пустое разнесенное развертывание, в которое вы сможете добавлять контент. Параметр «empty content» необходим для проверки того, что вы действительно намерены создать пустое развертывание, а не просто забыли определить содержимое.

```
[domain@localhost:9990 /] /deployment=kitchensink.ear:explode()
```

Это приведет к «разрыву» существующего архивного развертывания до его разнесенного формата. Эта операция не является рекурсивной, поэтому вам необходимо разорвать вложенное развертывание, если вы хотите иметь возможность манипулировать содержимым вложенного развертывания. Вы можете сделать это, указав путь к архиву вложенного развертывания в качестве параметра операции разнесения.

```
[domain@localhost:9990 /] /deployment=exploded.war:add-
content(content=[{targetpath=
WEBINF/
classes/org/jboss/as/test/deployment/trivial/ServiceActivatorDeployment
.class,
input-streamindex=/
home/demo/org/jboss/as/test/deployment/trivial/ServiceActivatorDeployme
nt.class
}, {target-path=META-INF/MANIFEST.MF, input-stream-
index=/home/demo/METAINF/
MANIFEST.MF}, {target-path=METAINF/
services/org.jboss.msc.service.ServiceActivator, input-streamindex=/
home/demo/META-INF/services/org.jboss.msc.service.ServiceActivator}])
```

Для каждого содержимого указывается исходное содержимое и целевой путь, в который оно будет скопировано относительно корневого каталога развертывания. В WildBoss Pro 11 вы можете использовать «input-stream-index» (который был удобным способом передачи потока содержимого) из CLI, указав его на локальный файл.

```
[domain@localhost:9990 /] /deployment=exploded.war:remove-
content(paths=[WEBINF/
classes/org/jboss/as/test/deployment/trivial/ServiceActivatorDeployment
.class,
META-INF/MANIFEST.MF, META-
INF/services/org.jboss.msc.service.ServiceActivator])
```

Теперь вы можете просмотреть содержимое развернутого развертывания или только его часть.

```
[domain@localhost:9990 /] /deployment=kitchensink.ear:browse-
content(archive=false,
path=WildBoss Pro-kitchensink-ear-web.war)
{
 "outcome" => "success",
 "result" => [
 {
   "path" => "META-INF/",
   "directory" => true
 },
 {
   "path" => "META-INF/MANIFEST.MF",
```

```
    "directory" => false,
    "file-size" => 128L
  },
  {
    "path" => "WEB-INF/",
    "directory" => true
  },
  {
    "path" => "WEB-INF/templates/",
    "directory" => true
  },
  {
    "path" => "WEB-INF/classes/",
    "directory" => true
  },
  {
    "path" => "WEB-INF/classes/org/",
    "directory" => true
  },
  {
    "path" => "WEB-INF/classes/org/jboss/",
    "directory" => true
  },
  {
    "path" => "WEB-INF/classes/org/jboss/as/",
    "directory" => true
  },
  {
    "path" => "WEB-INF/classes/org/jboss/as/quickstarts/",
    "directory" => true
  },
  {
    "path" => "WEB-
    INF/classes/org/jboss/as/quickstarts/kitchensink_ear/",
    "directory" => true
  },
  {
    "path" => "WEB-
    INF/classes/org/jboss/as/quickstarts/kitchensink_ear/controller/",
    "directory" => true
  },
  {
    "path" => "WEBINF/
    classes/org/jboss/as/quickstarts/kitchensink_ear/rest/",
    "directory" => true
  },
  {
    "path" => "WEBINF/
    classes/org/jboss/as/quickstarts/kitchensink_ear/util/",
    "directory" => true
  },
  {
    "path" => "resources/",
    "directory" => true
  },
  {
    "path" => "resources/css/",
    "directory" => true
```

```
  },
  {
    "path" => "resources/gfx/",
    "directory" => true
  },
  {
    "path" => "WEB-INF/templates/default.xhtml",
    "directory" => false,
    "file-size" => 2113L
  },
  {
    "path" => "WEB-INF/faces-config.xml",
    "directory" => false,
    "file-size" => 1365L
  },
  {
    "path" => "WEBINF/
    classes/org/jboss/as/quickstarts/kitchensink_ear/controller/MemberC
    ontroller.class
    ",
    "directory" => false,
    "file-size" => 2750L
  },
  {
    "path" => "WEBINF/
    classes/org/jboss/as/quickstarts/kitchensink_ear/rest/MemberResourc
    eRESTService.cl
    ass",
    "directory" => false,
    "file-size" => 6363L
  },
  {
    "path" => "WEB-
    INF/classes/org/jboss/as/quickstarts/kitchensink_ear/rest/JaxRsActi
    vator.class",
    "directory" => false,
    "file-size" => 464L
  },
  {
    "path" => "WEBINF/
    classes/org/jboss/as/quickstarts/kitchensink_ear/util/WebResources.
    class",
    "directory" => false,
    "file-size" => 667L
  },
  {
    "path" => "WEB-INF/beans.xml",
    "directory" => false,
    "file-size" => 1262L
  },
  {
    "path" => "index.xhtml",
    "directory" => false,
    "file-size" => 3603L
  },
  {
    "path" => "index.html",
    "directory" => false,
```

```
    "file-size" => 949L
  },
  {
    "path" => "resources/css/screen.css",
    "directory" => false,
    "file-size" => 4025L
  },
  {
    "path" => "resources/gfx/headerbkg.png",
    "directory" => false,
    "file-size" => 1147L
  },
  {
    "path" => "resources/gfx/asidebkg.png",
    "directory" => false,
    "file-size" => 1374L
  },
  {
    "path" => "resources/gfx/banner.png",
    "directory" => false,
    "file-size" => 41473L
  },
  {
    "path" => "resources/gfx/bkg-blkheader.png",
    "directory" => false,
    "file-size" => 116L
  },
  {
    "path" => "resources/gfx/rhjb_eap_logo.png",
    "directory" => false,
    "file-size" => 2637L
  },
  {
    "path" => "META-INF/maven/",
    "directory" => true
  },
  {
    "path" => "META-INF/maven/org.WildBoss Pro.quickstarts/",
    "directory" => true
  },
  {
    "path" => "META-INF/maven/org.WildBoss Pro.quickstarts/WildBoss
    Pro-kitchensink-earweb/",
    "directory" => true
  },
  {
    "path" => "META-INF/maven/org.WildBoss Pro.quickstarts/WildBoss
    Pro-kitchensink-earweb/
    pom.xml",
    "directory" => false,
    "file-size" => 4128L
  },
  {
    "path" => "META-INF/maven/org.WildBoss Pro.quickstarts/WildBoss
    Pro-kitchensink-earweb/
    pom.properties",
    "directory" => false,
    "file-size" => 146L
```

```
  }
  ]
}
```

У вас также есть операция чтения содержимого, но поскольку она возвращает двоичный поток, это невозможно отобразить из командной строки.

```
[domain@localhost:9990 /] /deployment=kitchensink.ear:read-
content(path=METAINF/
MANIFEST.MF)
{
  "outcome" => "success",
  "result" => {"uuid" => "b373d587-72ee-4b1e-a02a-71fbb0c85d32"},
  "response-headers" => {"attached-streams" => [{
  "uuid" => "b373d587-72ee-4b1e-a02a-71fbb0c85d32",
  "mime-type" => "text/plain"
  }]}
}
```

Однако интерфейс управления предоставляет высокоуровневые команды для отображения или сохранения вложений двоичного потока:

```
[domain@localhost:9990 /] attachment display
--operation=/deployment=kitchensink.ear:read-content(path=META-
INF/MANIFEST.MF)
ATTACHMENT d052340a-abb7-4a66-aa24-4eeeb6b256be:
Manifest-Version: 1.0
Archiver-Version: Plexus Archiver
Built-By: mjurc
Created-By: Apache Maven 3.3.9
Build-Jdk: 1.8.0_91
```

```
[domain@localhost:9990 /] attachment save --
operation=/deployment=kitchensink.ear:read
-content(path=META-INF/MANIFEST.MF) --file=example
File saved to /home/mjurc/WildBoss Pro/build/target/WildBoss Pro-
11.0.0.Alpha1-SNAPSHOT/example
```

### 6.1.3 XML-файл конфигурации

При развертывании содержимого контроллер домена добавляет в базу данных два типа записей. domain.xml файл конфигурации, один из которых содержит глобальную информацию о развертывании, а другой - для каждой соответствующей группы серверов, показывающий, как он используется этой группой серверов:

```
[...]
<deployments>
  <deployment name="test-application.war"
     runtime-name="test-application.war">
     <content sha1="dda9881fa7811b22f1424b4c5acccb13c71202bd"/>
  </deployment>
</deployments>
```

```
[...]
<server-groups>
  <server-group name="main-server-group" profile="default">
     [...]
     <deployments>
        <deployment name="test-application.war" runtime-name="test-application.war
        "/>
     </deployments>
  </server-group>
</server-groups>
[...]
```

(Смотри «domain/configuration/domain.xml»)

### 6.2 Автономный сервер

Развертывания на автономном сервере работают аналогично развертываниям в управляемых доменах. Основное отличие заключается в том, что здесь нет привязки к группам серверов.

#### 6.2.1 Команды развертывания

Те же команды CLI, которые используются для управляемых доменов, работают на автономных серверах при развертывании и удалении приложения:

```
[standalone@localhost:9990 /] deploy ~/Desktop/test-application.war
'test-application.war' deployed successfully.
[standalone@localhost:9990 /] undeploy test-application.war
Successfully undeployed test-application.war.
```

#### 6.2.2. Развертывание с помощью сканера развертывания

Содержимое для развертывания (например, файлы «war», «ear», «jar» и «sar») может быть помещено в автономный каталог «/deployments» дистрибутива WildBoss Pro для автоматического развертывания в среде выполнения сервера. Для этого должна присутствовать подсистема сканирования развертывания. Сканер периодически проверяет содержимое каталога развертываний и реагирует на изменения, обновляя сервер.

Внимание: пользователям рекомендуется использовать API-интерфейсы управления WildBoss Pro для загрузки и развертывания содержимого для развертывания, а не полагаться на сканер развертывания, который периодически сканирует каталог, особенно если используются производственные системы.

##### 6.2.2.1 Режимы развертывания сканера

Сканер развертывания файловой системы WildBoss Pro работает в одном из двух различных режимов, в зависимости от того, будет ли он непосредственно отслеживать содержимое развертываемой системы, чтобы принять решение о ее развертывании или повторном развертывании.

***Режим автоматического развертывания.***

Сканер будет напрямую отслеживать содержимое развертывания, автоматически развертывая новое содержимое и повторно размещая содержимое, у которого изменилась временная метка. Это аналогично поведению предыдущих версий AS, хотя есть различия:

– изменение любого файла в разнесенном развертывании запускает повторное развертывание. Поскольку приложениям «EE 6+» не требуются дескрипторы развертывания, не предпринимается попыток отслеживать дескрипторы развертывания и повторное развертывание выполняется только при изменении дескриптора развертывания;

– сканер поместит файлы-маркеры в этот каталог в качестве индикатора статуса попыток развертывания или отмены развертывания содержимого. Подробная информация о них приведена ниже.

***Режим развертывания вручную.***

Сканер не будет пытаться напрямую отслеживать содержимое развертываемого устройства и принимать решение о том, желает ли конечный пользователь, чтобы содержимое было развернуто, и когда именно. Вместо этого сканер использует систему файлов-маркеров, при этом добавление или удаление пользователем файла-маркера служит своего рода командой, указывающей сканеру на развертывание, отмену развертывания или повторное развертывание содержимого.

Автоматическое развертывание режим и режим ручного развертывания может быть независимо настроен на молнии содержание развертывания и взорвался развертывания контента. Это делается через опцию «автоматическое развертывание атрибут» на развертывание-сканер элемент в файле конфигурации «standalone.xml»:

```
<deployment-scanner scan-interval="5000" relative-
to="jboss.server.base.dir"
 path="deployments" auto-deploy-zipped="true" auto-deploy-
 exploded="false"/>
```

По умолчанию автоматическое развертывание архивированного содержимого включено, а автоматическое развертывание разнесенного содержимого отключено. Настоятельно рекомендуется использовать режим ручного развертывания для разнесенного содержимого, поскольку разнесенное содержимое по своей сути уязвимо для сканера, пытающегося автоматически развернуть частично скопированное содержимое.

#### 6.2.2.2 Файлы маркеров

Файлы-маркеры всегда имеют то же имя, что и содержимое развертывания, к которому они относятся, но с добавлением дополнительного файлового суффикса. Например, файл-маркер, указывающий, какой файл «example.war» должен быть развернут, называется «example.war.dodeploy». Различные суффиксы файлов-маркеров имеют разное значение.

Соответствующие типы файлов-маркеров приведены в таблице 3.

Таблица 3 – Типы файлов-маркеров

| Файл | Назначение |
|---|---|
| .dodeploy | размещается пользователем, чтобы указать, что данный контент должен быть развернут во время выполнения (или повторно развернут, если он уже был развернут во время выполнения) |

| Файл | Назначение |
| --- | --- |
| .skipdeploy | отключает автоматическое развертывание содержимого до тех пор, пока присутствует файл. Наиболее полезно для разрешения обновлений в «explodedcontent» без необходимости повторного развертывания сканером во время обновления. Можно использовать и с архивированным содержимым, хотя сканер обнаружит текущие изменения в архивированном содержимом и дождется завершения изменений |
| .isdeploying | размещается службой проверки развертывания, чтобы указать, что она обнаружила файл «.dodeploy» или новое или обновленное содержимое в режиме автоматического развертывания и находится в процессе развертывания содержимого. Этот файл-маркер будет удален после завершения процесса развертывания |
| .deployed | размещается службой проверки развертывания, чтобы указать, что данное содержимое было развернуто в среде выполнения. Если конечный пользователь удалит этот файл, содержимое не будет развернуто |
| .failed | размещается службой проверки развертывания, чтобы указать, что предоставленный контент не удалось развернуть во время выполнения. Содержимое файла будет содержать некоторую информацию о причине сбоя. Обратите внимание, что в режиме автоматического развертывания удаление этого файла сделает развертывание пригодным для повторного развертывания |
| .isundeploying | размещается службой проверки развертывания, чтобы указать, что было замечено, что развернутый файл был удален, а содержимое не развертывается. Этот файл-маркер будет удален после завершения процесса не развертывания |
| .undeployed | размещается службой проверки развертывания, чтобы указать, что предоставленное содержимое не было развернуто из среды выполнения. Если конечный пользователь удалит этот файл, это никак не повлияет |
| .pending | размещается службой сканирования развертывания, чтобы указать, что она |

| Файл | Назначение |
|---|---|
| | заметила необходимость развертывания содержимого, но еще не настроила сервер для его развертывания. Этот файл создается, если сканер обнаруживает, что какой-либо автоматически развертываемый контент все еще находится в процессе копирования, или если возникает какая -либо проблема, препятствующая автоматическому развертыванию. Сканер не будет указывать серверу на развертывание или отмену развертывания какого-либо контента (не только непосредственно затронутого контента), пока выполняется это условие |

***Основные рабочие процессы.***

Во всех примерах предполагается, что переменная «$JBOSS_HOME» указывает на корень распределения WildBoss Pro.

1) добавьте новый архивированный контент и разверните его:

– цель «cp/example.war/ $JBOSS_HOME/standalone/deployments»;

– коснитесь «$JBOSS_HOME/standalone/deployments/example.war.dodeploy» (только в ручном режиме);

2) добавьте новый распакованный контент и разверните его:

– цель «cp/example.war/ $JBOSS_HOME/standalone/deployments»;

– коснитесь «$JBOSS_HOME/standalone/deployments/example.war.dodeploy» (только в ручном режиме);

3) отмените развертывание текущего содержимого:

– «rm $JBOSS_HOME/standalone/deployments/example.war.deployed»;

4) только режим автоматического развертывания «Отмените развертывание текущего содержимого»:

– «rm $JBOSS_HOME/standalone/deployments/example.war»;

5) замените развернутый в данный момент архивированный контент новой версией и разверните его:

– цель «cp/example.war/ $JBOSS_HOME/standalone/deployments»;

– коснитесь «$JBOSS_HOME/standalone/deployments/example.war.dodeploy» (только в ручном режиме);

6) только в ручном режиме: замените развернутый в данный момент распакованный контент новой версией и разверните его:

– «rm $JBOSS_HOME/standalone/deployments/example.war.deployed»;

– дождитесь появления файла «$JBOSS_HOME/standalone/deployments/example.war.undeployed»;

– цель «cp -r /example.war/ $JBOSS_HOME/standalone/deployments»;

– коснитесь «$JBOSS_HOME/standalone/deployments/example.war.dodeploy»;

7) только режим автоматического развертывания: замените развернутое в данный момент распакованное содержимое новой версией и разверните его:

– нажмите «$JBOSS_HOME/standalone/deployments/example.war.skipdeploy»;

– цель «cp -r/example.war/ $JBOSS_HOME/standalone/deployments»;

– «rm $JBOSS_HOME/standalone/deployments/example.war.skipdeploy»;

8) только ручной режим: оперативная замена частей развернутого в данный момент разархивированного содержимого без повторного развертывания:

– цель «cp-r/example.war/foo.html$JBOSS_HOME/standalone/deployments/ example.war»;

9) только режим автоматического развертывания «Оперативная замена частей развернутого в данный момент разархивированного содержимого без повторного развертывания»:

– нажмите «$JBOSS_HOME/standalone/deployments/example.war.skipdeploy»;

– цель «cp-r/example.war/foo.html$JBOSS_HOME/standalone/deployments/ example.war»;

10) ручной или автоматический режим развертывания «Повторно разверните текущее содержимое (т.е. верните его без изменения содержимого)»:

– коснитесь «$JBOSS_HOME/standalone/deployments/example.war.dodeploy»;

11) только режим автоматического развертывания: повторно разверните текущее содержимое (т.е. верните его без изменения содержимого):

– коснитесь «$JBOSS_HOME/standalone/deployments/example.war».

**Внимание:** в приведенных выше примерах используются команды оболочки Unix. Эквивалентами Windows являются:

cp src
dest -⟶ xcopy /y src dest
cp -r src dest -⟶ xcopy /e /s /y src dest
rm afile -⟶ del afile
нажмите afile -⟶ echo>> afile

Обратите внимание, что поведение «touch» и «echo» различно, но эти различия не имеют отношения к использованию в приведенных выше примерах.

### 6.3 Управляемые и неуправляемые развертывания

WildBoss Pro поддерживает два механизма работы с контентом развертывания – управляемое и неуправляемое развертывание.

При управляемом развертывании сервер получает содержимое развертывания и копирует его во внутреннее хранилище контента, а затем использует эту копию контента, а не исходный контент, предоставленный пользователем. После этого сервер несет ответственность за содержимое, которое он использует.

При неуправляемом развертывании пользователь указывает путь к размещаемому содержимому в локальной файловой системе, и сервер напрямую использует это содержимое. Однако пользователь несет ответственность за сохранность содержимого, например, за то, чтобы в него не вносились изменения, которые негативно повлияли бы на функционирование развернутого приложения.

Чтобы помочь вам отличить управляемое развертывание от неуправляемого, модель развертывания имеет логический атрибут среды выполнения «managed».

Управляемые развертывания имеют ряд преимуществ перед неуправляемыми:

– клиенты удаленного управления могут управлять ими, не требуя доступа к файловой системе сервера;

– в управляемом домене WildBoss Pro /EAP возьмет на себя ответственность за репликацию копии развертывания на все хосты/серверы в домене, где это необходимо. При неуправляемом развертывании пользователь несет ответственность за то, чтобы развертывание было доступно в локальной файловой системе на всех соответствующих хостах по согласованному пути;

– фактически используемое содержимое развертывания хранится в файловой системе во внутреннем хранилище содержимого, что должно помочь защитить его от непреднамеренных изменений.

Все предыдущие примеры, приведенные выше, иллюстрируют использование управляемых развертываний, за исключением случаев, когда речь идет об обработке

разнесенных развертываний сканером развертывания. В WildBoss Pro 10 и более ранних версиях разнесенные развертывания всегда неуправляемы, начиная с WildBoss Pro 11 это уже не так.

### 6.3.1 Хранилище контента

В случае управляемого развертывания фактический файл, используемый сервером при создании служб среды выполнения, не является файлом, предоставляемым команде CLI «deploy» или веб-консоли. Это копия этого файла, хранящаяся во внутреннем хранилище содержимого. Хранилище находится в каталоге «domain/data/content» для управляемого домена или в каталоге «standalone/data/content» для автономного сервера. Сами двоичные файлы хранятся в подкаталоге:

```
ls domain/data/content/
  |---/47
  |-----95cc29338b5049e238941231b36b3946952991
  |---/dd
  |-----a9881fa7811b22f1424b4c5acccb13c71202bd
```

**Внимание:** местоположение хранилища контента и его внутренняя структура могут быть изменены в любое время, и конечным пользователям не следует полагаться на них.

Описание управляемого развертывания в доменном или автономном файле конфигурации включает атрибут, записывающий хэш «SHA1» содержимого развертывания:

```
<deployments>
   <deployment name="test-application.war"
               runtime-name="test-application.war">
      <content sha1="dda9881fa7811b22f1424b4c5acccb13c71202bd"/>
   </deployment>
</deployments>
```

Процесс WildBoss Pro вычисляет и записывает этот хэш, когда пользователь запускает операцию управления (например, команду CLI «deploy» или с помощью консоли), предоставляющую содержимое для развертывания. Предполагается, что пользователь не будет вычислять хэш.

Атрибут «sha1» в элементе «content» указывает процессу WildBoss Pro, где найти контент для развертывания в его внутреннем хранилище контента.

В домене каждый хост будет иметь копию контента, необходимого его серверам, в своем собственном локальном хранилище контента. Процессы WildBoss Pro «domain controller» и «slave host controller» берут на себя ответственность за обеспечение каждого хоста необходимым контентом.

### 6.3.2 Неуправляемое развертывание

Неуправляемое развертывание - это процесс, при котором сервер напрямую развертывает содержимое по указанному вами пути вместо создания внутренней копии и последующего развертывания копии.

Первоначальное развертывание неуправляемого развертывания во многом похоже на развертывание управляемого, за исключением того, что вы сообщаете WildBoss Pro, что не хотите, чтобы развертывание было управляемым:

```
[standalone@localhost:9990 /] deploy ~/Desktop/test-application.war --
unmanaged
'test-application.war' deployed successfully.
```

Когда вы это сделаете, вместо того, чтобы сервер создавал копию содержимого в «/Desktop/testapplication.war», вычисляя хэш содержимого, сохраняя хэш в файле конфигурации и затем устанавливая копию в среду выполнения, вместо этого он преобразует «/Desktop/test-application.war» укажите абсолютный путь, сохраните его в файле конфигурации, а затем установите исходное содержимое во время выполнения.

Вы также можете использовать неуправляемые развертывания в домене:

```
[domain@localhost:9990 /] deploy /home/example/Desktop/test-
application.war --server
-group=main-server-group --unmanaged
'test-application.war' deployed successfully.
```

Однако перед выполнением этой команды необходимо убедиться, что копия содержимого присутствует на всех компьютерах, на которых установлены серверы из целевых групп серверов, и все они имеют одинаковый путь к файловой системе. Домен не будет копировать файл для вас.

Отмена развертывания ничем не отличается от управляемой отмены развертывания:

```
[standalone@localhost:9990 /] undeploy test-application.war
Successfully undeployed test-application.war.
```

Выполнение замены развертывания на новую версию немного отличается, сервер использует файл, который вы хотите заменить. Вам следует отменить развертывание, заменить содержимое и затем выполнить повторное развертывание. Или вы можете остановить сервер, заменить развертывание и выполнить повторное развертывание.

## 6.4 Наложения при развертывании

Наложения при развертывании - это наш способ «наложения» содержимого на существующее развертывание без физического изменения содержимого архива развертывания. Возможные варианты использования включают замену дескрипторов развертывания, изменение статических веб-ресурсов для изменения фирменного стиля приложения или даже замену библиотек «jar» на другие версии.

Жизненный цикл оверлеев развертывания отличается от жизненного цикла развертывания. Чтобы использовать оверлей для развертывания, вы сначала создаете оверлей, используя интерфейс командной строки или API управления. Затем вы добавляете файлы в оверлей, указывая пути развертывания, которые вы хотите, чтобы они перекрывались. После того, как вы создали наложение, вам необходимо привязать его к имени развертывания (это делается немного по-разному в зависимости от того, работаете ли вы в автономном режиме или в режиме домена). После создания ссылки наложение будет применено к любому развертыванию, которое соответствует указанному имени развертывания.

Когда вы изменяете или создаете наложение, это не влияет на существующие развертывания, их необходимо повторно развернуть.

### 6.4.1 Создание наложения для развертывания

Для создания наложения развертывания интерфейс командной строки предоставляет команду высокого уровня, позволяющую выполнить все указанные выше действия за один раз. Ниже приведен пример команды как для автономного, так и для доменного режима:

```
deployment-overlay add --name=myOverlay --content=/WEB
-INF/web.xml=/myFiles/myWeb.xml,/WEB-INF/ejb-
jar.xml=/myFiles/myEjbJar.xml
--deployments=test.war,*-admin.war --redeploy-affected
```

```
deployment-overlay add --name=myOverlay --content=/WEB
-INF/web.xml=/myFiles/myWeb.xml,/WEB-INF/ejb-
jar.xml=/myFiles/myEjbJar.xml
--deployments=test.war,*-admin.war --server-groups=main-server-group -
-redeploy
-affected
```

# 7 Конфигурация подсистемы

Ссылка на конфигурацию подсистемы :автор: tcerar@redhat.com :значки: шрифт :источник-маркер: кодировка :оглавление: макросы :уровни: 2.

В следующих главах будут рассмотрены варианты использования высокоуровневого управления, доступные через CLI и веб-интерфейс. Для получения подробного описания каждого свойства конфигурации подсистемы, пожалуйста, обратитесь к соответствующему справочнику компонентов.

Расположение в схеме.

**Внимание:** схемы конфигурации можно найти в файле «$JBOSS_HOME/docs/schema».

## 7.1 Конфигурация подсистемы EE

Подсистема EE обеспечивает общие функциональные возможности платформы Jakarta EE, такие как EE Утилиты параллелизма (JSR 236) и внедрение @Resource. Подсистема также отвечает за управление жизненным циклом развертываний приложений Jakarta EE, то есть за файлы «*.ear» и настройку глобальных каталогов для совместного использования общих библиотек во всех развернутых приложениях.

Конфигурация подсистемы EE может быть использована для:

– настройте развертывание приложений Jakarta EE по индивидуальному заказу;

– создание экземпляров утилит параллелизма EE;

– определите привязки по умолчанию.

Имя подсистемы – «ee», и этот документ описывает версию 5.0 подсистемы EE, пространство имен XML которой в конфигурациях WildBoss Pro XML имеет значение «urn:jboss:domain:ee:5.0». Путь к XML-схеме подсистемы в дистрибутиве WildBoss Pro - это «docs/schema/jboss-as-ee_5_0.xsd».

Пример конфигурации подсистемы в формате XML со всеми указанными элементами и атрибутами:

```
<subsystem xmlns="urn:jboss:domain:ee:5.0">
 <global-modules>
   <module name="org.jboss.logging"
           slot="main"/>
   <module name="org.apache.log4j"
           annotations="true"
           meta-inf="true"
           services="false" />
 </global-modules>
 <global-directories>
   <directory name="common-libs" path="libs" relative-
   to="jboss.server.base.dir
 "/>
 </global-directories>
 <ear-subdeployments-isolated>true</ear-subdeployments-isolated>
 <spec-descriptor-property-replacement>false</spec-descriptor-
 property-replacement>
 <jboss-descriptor-property-replacement>false</jboss-descriptor-
 property-
replacement>
  <annotation-property-replacement>false</annotation-property-
  replacement>
  <concurrent>
   <context-services>
    <context-service
      name="default"
```

```
        jndi-name="java:jboss/ee/concurrency/context/default"
        use-transaction-setup-provider="true" />
    </context-services>
    <managed-thread-factories>0
     <managed-thread-factory
        name="default"
        jndi-name="java:jboss/ee/concurrency/factory/default"
        context-service="default"
        priority="1" />
    </managed-thread-factories>
    <managed-executor-services>
     <managed-executor-service
        name="default"
        jndi-name="java:jboss/ee/concurrency/executor/default"
        context-service="default"
        thread-factory="default"
        hung-task-threshold="60000"
        core-threads="5"
        max-threads="25"
        keepalive-time="5000"
        queue-length="1000000"
        reject-policy="RETRY_ABORT" />
    </managed-executor-services>
    <managed-scheduled-executor-services>
     <managed-scheduled-executor-service
        name="default"
        jndi-name="java:jboss/ee/concurrency/scheduler/default"
        context-service="default"
        thread-factory="default"
        hung-task-threshold="60000"
        core-threads="5"
        keepalive-time="5000"
        reject-policy="RETRY_ABORT" />
    </managed-scheduled-executor-services>
  </concurrent>
  <default-bindings
       context-service="java:jboss/ee/concurrency/context/default"
       datasource="java:jboss/datasources/ExampleDS"
       jms-connection-factory="java:jboss/DefaultJMSConnectionFactory"
       managed-executor-
       service="java:jboss/ee/concurrency/executor/default"
       managed-scheduled-executor-service=
       "java:jboss/ee/concurrency/scheduler/default"
       managed-thread-
       factory="java:jboss/ee/concurrency/factory/default" />
</subsystem>
```

### 7.1.1 Внедрение приложений EE в Jakarta

Конфигурация подсистемы EE позволяет настраивать поведение при развертывании для Jakarta приложения EE.

#### 7.1.1.1 Глобальные модули

Global modules - это набор модулей JBoss, которые будут добавлены в качестве зависимостей к модулю JBoss Modules каждого развертывания Jakarta EE. Такие зависимости

позволяют развертываниям Jakarta EE видеть классы, экспортируемые глобальными модулями.

Каждый глобальный модуль определяется с помощью ресурса «module», примера его XML-конфигурации:

```
<global-modules>
  <module name="org.jboss.logging" slot="main"/>
  <module name="org.apache.log4j" annotations="true" meta-inf="true"
  services="
  false" />
</global-modules>
```

Единственным обязательным атрибутом является имя модуля («name») JBoss Modules, атрибут «slot» по умолчанию имеет значение «main», и оба они определяют идентификатор модуля JBoss Modules для ссылки.

Необязательный атрибут «annotations», значение которого по умолчанию равно «false», указывает, следует ли импортировать предварительно вычисленный индекс аннотации из «META-INF/jandex.idx».

Атрибут необязательных служб указывает, должны ли какие-либо службы, представленные в «META-INF/services», быть доступны для загрузчика классов развертываний, и по умолчанию имеет значение «false».

Необязательный атрибут «meta-inf», значение которого по умолчанию равно «true», указывает, должен ли путь к META-INF модуля быть доступен для загрузчика классов при развертывании.

#### 7.1.1.2 Глобальный каталог

Глобальные модули можно использовать для предоставления общего доступа к библиотекам во всех развернутых приложениях, но это может оказаться непрактичным, если имя общей библиотеки меняется очень часто или если вы хотите предоставить общий доступ к большому количеству библиотек. В обоих случаях потребуются изменения в базовом «module.xml», который представляет этот глобальный модуль.

Подсистема EE позволяет настраивать глобальный каталог, который представляет собой дерево каталогов, автоматически сканируемое для «application. Basically» и ресурсы «jar» являются единой дополнительной зависимостью. Эта зависимость от модуля добавляется как системная зависимость к каждому развернутому приложению. По сути, при работе с глобальным каталогом вы будете полагаться на WildBoss Pro для автоматизации обслуживания и настройки модуля JBoss Modules, который представляет jar-файлы и ресурсы определенного каталога.

Вы можете настроить глобальный каталог, выполнив следующую операцию:

```
[standalone@localhost:9990 /] /subsystem=ee/global-directory=my-
commonlibs:
add(path=lib, relative-to=jboss.home.dir)
```

На ресурсе «global-directory» доступны следующие атрибуты:

– path: путь к каталогу для сканирования (обязательно);

– relative-to: имя другого ранее названного пути или одного из стандартных путей , предоставляемых системой. (опционально).

Когда создается глобальный каталог, сервер устанавливает модуль JBoss Modules с одним путем Загрузчик ресурсов, созданный с использованием атрибутов «path» и «relative-

to», и один загрузчик ресурсов «Jar» для каждого файла «jar», включенного в этот каталог и его подкаталоги.

«Загрузчик ресурсов Path» сделает доступным для приложения любой файл в качестве ресурса. «Загрузчик ресурсов Jar» сделает доступным для приложений любой класс внутри файла «jar».

Например, предположим, что вы настроили один глобальный каталог, указывающий на следующее дерево каталогов:

```
/my-common-libs/Z/a-lib.jar
/my-common-libs/A/A/z-lib.jar
/my-common-libs/A/a-lib.jar
/my-common-libs/A/b-lib.jar
/my-common-libs/a-lib.jar
/my-common-libs/A/B/a-lib.jar
/my-common-libs/properties-1.properties
/my-common-libs/A/B/properties-2.properties
```

```
<module xmlns="urn:jboss:module:1.9" name="deployment.external.global-
directory.mycommon-
libs">
    <resources>
      <resource-root path="/my-common-libs"/>
      <resource-root path="/my-common-libs/a-lib.jar"/>
      <resource-root path="/my-common-libs/A/a-lib.jar"/>
      <resource-root path="/my-common-libs/A/b-lib.jar"/>
      <resource-root path="/my-common-libs/A/A/z-lib.jar"/>
      <resource-root path="/my-common-libs/A/B/a-lib.jar"/>
      <resource-root path="/my-common-libs/Z/a-lib.jar"/>
    </resources>
    <dependencies>
    <module name="javaee.api"/>
    </dependencies>
</module>
```

Имя сгенерированного модуля соответствует шаблону «*deployment.external.global-directory.{globaldirectory-name}»,* и поэтому его можно выборочно исключить, используя ваш «`deployment-structure.xml`».

Все ресурсы будут доступны из загрузчика классов приложения. Например, вы могли бы получить доступ к указанным выше файлам свойств, используя контекстный загрузчик классов вашего текущего потока:

```
Thread.currentThread().getContextClassLoader().getResourceAsStream("pro
perties-
1.properties");
Thread.currentThread().getContextClassLoader().getResourceAsStream("A/B
/properties-
2.properties");
```

Все классы внутри каждого jar-файла также будут доступны, и порядок внутреннего создания корневых ресурсов будет определять порядок загрузки классов. Ресурсы «jar» сгенерированного модуля будут создаваться итерационно по всем jar-файлам, найденным в дереве каталогов. Каждый каталог просматривается в алфавитном порядке, начиная с корневого, и на каждом уровне каждый подкаталог также просматривается в алфавитном порядке, пока не будут пройдены все ветви. Файлы, найденные на каждом уровне, также добавляются в алфавитном порядке.

Обратите внимание, что вы должны знать, какие классы доступны для каждого файла «*.jar», и избегать конфликтов, в том числе использования одного и того же класса дважды с несовместимыми двоичными изменениями. В таких случаях, вероятно, возникают ошибки при загрузке классов. В частности, вам не следует добавлять классы, которые мешают классам, которые сервер уже предоставляет вашему приложению; цель глобального каталога - не переопределять и не заменять существующие версии библиотек, поставляемые вместе с сервером. Это средство, которое позволит перенести общие фреймворки, которые вы обычно добавляете в свои библиотеки приложений, в общее место для облегчения обслуживания.

Модуль, созданный из общего каталога, загружается, как только на сервере развертывается первое приложение после создания глобального каталога. Это означает, что если сервер запущен/перезапущен, а приложения не развернуты, то глобальный каталог не сканируется и модуль не загружается. Любое изменение содержимого глобального каталога потребует перезагрузки сервера, чтобы сделать его доступным для развернутых приложений.

В случае доменного режима или распределенных сред пользователь несет ответственность за согласование содержимого настроенного глобального каталога во всех экземплярах сервера, а также за распространение файлов «*.jar», которые они содержат.

#### 7.1.1.3 Изоляция вспомогательных функций EAR

Флаг, указывающий, может ли каждое из подразделений в «.ear» получать доступ к классам, принадлежащим другому подразделению в том же «.ear». Значение по умолчанию равно «`false`», что позволяет подразделениям видеть классы, принадлежащие другим подразделениям в «.ear».

```
<ear-subdeployments-isolated>true</ear-subdeployments-isolated>
```

Например:

```
myapp.ear
|
|--- web.war
|
|--- ejb1.jar
|
|--- ejb2.jar
```

Если параметру «`ear-subdeployments-isolated`» присвоено значение «`false`», то классы в «*web.war*» могут получать доступ к классам, принадлежащим «ejb1.jar» и «ejb2.jar». Аналогично, классы из «ejb1.jar» могут получать доступ к классам из «ejb2.jar» (и наоборот).

**Внимание:** этот флаг не влияет на изолированный загрузчик классов «.war»-файлов, т.е. независимо от того, установлено ли для этого флага значение «true» или «false», «.war» в «.ear» будет иметь изолированный загрузчик классов, и другие подразделения в этом .ear не смогут получить доступ к классам из этого «.war». Это соответствует спецификации.

#### 7.1.1.4 Замена имущества

Конфигурация подсистемы EE включает флаги для настройки того, будет ли производиться замена системных свойств в XML-дескрипторах и аннотациях Java, включенных в развертывания Jakarta EE.

**Внимание:** системные свойства и т.д. решаются в контексте безопасности самого сервера приложений, а не развертывания, содержащего файл. Это означает, что, если вы работаете с «security manager» и включаете это свойство, развертывание потенциально может

получить доступ к системным свойствам или записям среды, которые в противном случае были бы предотвращены «security manager».

***1) Замена свойства дескриптора спецификации***

Флажок, указывающий, будет ли выполняться замена системных свойств в стандартных Jakarta EE XML-дескрипторах. Если этот параметр не настроен, по умолчанию он равен «`true`», однако в стандартных файлах конфигурации, поставляемых с WildBoss Pro, для него установлено значение «`false`».

```
<spec-descriptor-property-replacement>false</spec-descriptor-property-
replacement>
```

Если этот параметр включен, свойства могут быть заменены в следующих дескрипторах развертывания:

– ejb-jar.xml;
– persistence.xml;
– application.xml;
– web.xml;
– permissions.xml.

***2) Замена свойства дескриптора JBoss***

Флаг, указывающий, будет ли выполнена замена системных свойств на WildBoss Pro proprietary XML-дескрипторы, такие как «jboss-app.xml». По умолчанию это значение равно «`true`».

```
<jboss-descriptor-property-replacement>false</jboss-descriptor-
property-replacement>
```

Если этот параметр включен, свойства могут быть заменены в следующих дескрипторах развертывания:

– jboss-ejb3.xml;
– jboss-app.xml;
– jboss-web.xml;
– jboss-permissions.xml;
– *-jms.xml;
– *-ds.xml.

***3) Замена свойства аннотации***

Флаг, указывающий, будет ли выполняться замена системных свойств в Java-аннотациях. Значение по умолчанию – «`false`».

### 7.1.2 Утилиты для обеспечения параллелизма EE

Утилиты EE Concurrency Utilities (JSR 236) были представлены для облегчения задачи написания многопоточных приложений. Экземпляры этих утилит управляются WildBoss Pro и соответствующей конфигурацией.

#### 7.1.2.1 Контекстные сервисы

Контекстная служба - это утилита параллелизма, которая создает контекстные прокси-серверы из существующих объектов. Контекстные службы WildBoss Pro также используются для распространения контекста из потока вызова приложения Jakarta EE в потоки, используемые другими утилитами параллелизма EE. Контекст Экземпляры служб могут быть созданы с использованием конфигурации XML подсистемы:

```
 <context-services>
   <context-service
name="default"
jndi-name="java:jboss/ee/concurrency/context/default"
use-transaction-setup-provider="true" />
 </context-services>
```

Атрибут «name» является обязательным, и его значением должно быть уникальное имя во всех контекстных службах.

Атрибут «jndi-name» также является обязательным и определяет, где в JNDI должна быть размещена контекстная служба.

Необязательный атрибут «use-trasaction-setup-provider» указывает, должны ли контекстные прокси, созданные контекстной службой, приостанавливать транзакции в контексте при вызове объектов прокси, и его значение по умолчанию равно «true».

Управляющие клиенты, такие как WildBoss Pro CLI, также могут использоваться для настройки экземпляров контекстной службы. Пример добавления и удаления одного из них с именем «other»:

```
/subsystem=ee/context-service=other:add(jndi-
         name=java\:jboss\/ee\/concurrency\/other)
/subsystem=ee/context-service=other:remove
```

#### 7.1.2.2 Управляемые Factory по производству потоков

Factory управляемых потоков позволяет приложениям Jakarta EE создавать новые потоки. WildBoss Pro экземпляры Factory управляемых потоков могут также, при необходимости, использовать экземпляр контекстной службы для распространения контекста потока приложения Jakarta EE на новые потоки. Создание экземпляра осуществляется через подсистему EE путем редактирования XML-конфигурации подсистемы:

```
 <managed-thread-factories>
   <managed-thread-factory
name="default"
jndi-name="java:jboss/ee/concurrency/factory/default"
context-service="default"
priority="1" />
 </managed-thread-factories>
```

Атрибут «name» является обязательным, и его значение должно быть уникальным именем во всех управляемых потоках Factory.

Атрибут «jndi-name» также является обязательным и определяет, где в JNDI должен быть размещен управляемый поток Factory.

Атрибут «jndi-name» также является обязательным и определяет, где в JNDI должен быть размещен управляемый поток Factory.

Необязательный «context-service» ссылается на существующий контекстный сервис по его «name». Если указано, то поток, созданный Factory, будет распространять контекст вызова, присутствующий при создании потока.

Необязательный «priority» указывает приоритет для новых потоков, созданных Factory, и по умолчанию равен 5.

Управляющие клиенты, такие как WildBoss Pro CLI, также могут использоваться для настройки управляемого потока Factory экземпляры. Пример добавления и удаления одного из них с именем «other»:

```
/subsystem=ee/managed-thread-factory=other:add(jndiname=
         Java\:jboss\/ee\/factory\/other)
/subsystem=ee/managed-thread-factory=other:remove
```

#### 7.1.2.3 Управляемые службы исполнителей

Служба Managed Executor - это адаптация Java SE Executor Service для Jakarta EE, предоставляющая приложениям Jakarta EE функциональность асинхронного выполнения задач. WildBoss Pro отвечает за управление жизненным циклом управляемых экземпляров службы Executor, которые задаются с помощью XML-конфигурации подсистемы EE:

```
<managed-executor-service
    name="default"
    jndi-name="java:jboss/ee/concurrency/executor/default"
    context-service="default"
    thread-factory="default"
    hung-task-threshold="60000"
    hung-task-termination-period="60000"
    core-threads="5"
    max-threads="25"
    keepalive-time="5000"
    queue-length="1000000"
    reject-policy="RETRY_ABORT" />
</managed-executor-services>
```

Атрибут «name» является обязательным, и его значением должно быть уникальное имя во всех управляемых службах Executor.

Атрибут «jndi-name» также является обязательным и определяет, где в JNDI должен быть размещен управляемый исполнитель Service.

Необязательная контекстная служба («context-service») ссылается на существующую контекстную службу по ее имени («name»). Если указано, то контекстная служба, на которую ссылается ссылка, будет фиксировать контекст вызова, присутствующий при отправке задачи исполнителю, который затем будет использоваться при выполнении задачи.

Обязательный параметр «core-threads» определяет количество потоков, которые должны сохраняться в пуле исполнителя, даже если они находятся в режиме ожидания. Если это значение не определено или равно 0, размер пула ядер будет рассчитываться на основе количества доступных процессоров.

Необязательная длина очереди («queue-length») указывает количество задач, которые могут быть сохранены во входной очереди. Значение по умолчанию равно 0, что означает, что объем очереди не ограничен.

Очередь задач исполнителя основана на значениях атрибутов «core-threads» (основные потоки) и «queue-length» (длина очереди):

– если длина очереди («queue-length») равна 0 или если длина очереди равна целому числу Integer.MAX_VALUE (2147483647) и число основных потоков равно 0, будет использована стратегия организации очереди с прямой передачей обслуживания и будет создана синхронная очередь;

– если длина очереди («queue-length») равна целому числу Integer.MAX_VALUE (2147483647), но для основных потоков не равно 0, будет использоваться неограниченная очередь;

– для любого другого допустимого значения длины очереди будет создана ограниченная очередь.

Необязательный параметр «hanged-task-threshold» определяет пороговое значение времени выполнения в миллисекундах, при котором задачи будут считаться зависшими для исполнителя. Значение 0 никогда не будет считать задачи зависшими.

Необязательный период завершения зависшей задачи («hanged-task-termination period») определяет период в миллисекундах, в течение которого можно попытаться завершить зависшие задачи, отменив их выполнение и прервав выполняющиеся потоки. Пожалуйста, обратите внимание, что завершение отмененной зависшей задачи не гарантируется. Значение 0, установленное по умолчанию, отключает периодическую отмену зависших задач. Управляющие клиенты, такие как WildBoss Pro, все еще могут использоваться для выполнения попыток вручную завершить зависшие задачи:

```
/subsystem=ee/managed-executor-service=other:terminate-hung-tasks
```

Необязательный параметр «long-running-tasks» - это подсказка для оптимизации выполнения длительных задач, значение по умолчанию равно «false».

Необязательный параметр «max-threads» определяет максимальное количество потоков, используемых исполнителем, которое по умолчанию равно целому числу Integer.MAX_VALUE (2147483647).

Необязательный параметр «keepalive-time» определяет время простоя внутреннего потока в миллисекундах. Значение атрибута по умолчанию равно 60000.

Необязательная политика отклонения определяет политику, которая будет использоваться, когда задача отклоняется исполнителем. Значением атрибута может быть значение «ABORT» по умолчанию, которое означает, что должно быть сгенерировано исключение, или «RETRY_ABORT», которое означает, что исполнитель попытается отправить его еще раз, прежде чем сгенерировать исключение.

Управляющие клиенты, такие как WildBoss Pro CLI, также могут использоваться для настройки управляемого Managed Executor Service instances. Пример добавления и удаления одного из них с именем «other»:

```
/subsystem=ee/managed-executor-service=other:add(jndiname=
        java\:jboss\/ee\/executor\/other, core-threads=2)
/subsystem=ee/managed-executor-service=other:remove
```

#### 7.1.2.4 Управляемые службы запланированного исполнителя

Служба управляемого запланированного исполнителя (Managed Scheduled Executor Service) - это адаптация Java SE для Jakarta EE по расписанию Служба Executor Service, предоставляющая приложениям Jakarta EE функциональность планирования выполнения задач. WildBoss Pro отвечает за управление жизненным циклом управляемых

запланированных экземпляров службы Executor, которые задаются в XML-конфигурации подсистемы EE:

```
<managed-scheduled-executor-services>
  <managed-scheduled-executor-service
     name="default"
     jndi-name="java:jboss/ee/concurrency/scheduler/default"
     context-service="default"
     thread-factory="default"
     hung-task-threshold="60000"
     core-threads="5"
     keepalive-time="5000"
     reject-policy="RETRY_ABORT" />
</managed-scheduled-executor-services>
```

Атрибут «name» является обязательным, и его значением должно быть уникальное имя во всех управляемых службах Scheduled Executor.

Атрибут «jndi-name» также является обязательным и определяет, где в JNDI должна быть размещена управляемая запланированная служба Executor.

Необязательная контекстная служба («context-service») ссылается на существующую контекстную службу по ее имени. Если указано, то контекстная служба, на которую ссылается ссылка, будет фиксировать контекст вызова, присутствующий при отправке задачи исполнителю, который затем будет использоваться при выполнении задачи.

Необязательная «thread-factory» ссылается на существующую фабрику управляемых потоков по своему имени, чтобы управлять созданием внутренних потоков. Если не указано иное, то будет создана фабрика управляемых потоков с конфигурацией по умолчанию и будет использоваться внутри компании.

Обязательный параметр «core-threads» определяет количество потоков, которые необходимо сохранить в пуле исполнителя, даже если они находятся в режиме ожидания. Значение 0 означает, что ограничений нет.

Необязательный параметр «hanged-task-threshold» определяет пороговое значение времени выполнения в миллисекундах, при котором задачи будут считаться зависшими для исполнителя. Значение 0 никогда не будет считать задачи зависшими.

Необязательный «hung-task-termination-period» определяет период в миллисекундах, в течение которого можно попытаться завершить зависшие задачи, отменив их выполнение и прервав выполняющиеся потоки. Пожалуйста, обратите внимание, что завершение отмененной зависшей задачи не гарантируется. Значение 0, установленное по умолчанию, отключает периодическую отмену зависших задач. Управляющие клиенты, такие как WildBoss Pro CLI, все еще могут использоваться для выполнения попыток завершения зависших задач вручную:

```
/subsystem=ee/managed-scheduled-executor-service=other:terminate-hung-
      tasks
```

Необязательный параметр «long-running-tasks» - это подсказка для оптимизации выполнения длительных задач, значение по умолчанию равно «false».

Необязательный параметр «keepalive-time» определяет время простоя внутреннего потока в миллисекундах. Значение атрибута по умолчанию равно 60000.

Необязательная политика отклонения определяет политику, которая будет использоваться, когда задача отклоняется исполнителем. Значением атрибута может быть значение «ABORT» по умолчанию, которое означает, что должно быть сгенерировано исключение, или «RETRY_ABORT», которое означает, что исполнитель попытается отправить его еще раз, прежде чем сгенерировать исключение.

Управляющие клиенты, такие как WildBoss Pro CLI, также могут использоваться для настройки управляемого расписания Экземпляры службы-исполнителя. Пример добавления и удаления одного из них с именем «other»:

```
/subsystem=ee/managed-scheduled-executor-service=other:add(jndiname=
      java\:jboss\/ee\/scheduler\/other, core-threads=2)
/subsystem=ee/managed-scheduled-executor-service=other:remove
```

### 7.1.3 Привязки EE по умолчанию

Спецификация Jakarta EE требует наличия экземпляра по умолчанию для каждого из следующих ресурсов:

– Context Service (Контекстная служба);

– Datasource (Источник данных);

– Jakarta Messaging Connection Factory (Фабрика по подключению к системе обмена сообщениями в Джакарте);

– Managed Executor Service (Служба управляемого исполнителя);

– Managed Scheduled Executor Service (Управляемая служба запланированного исполнителя);

– Managed Thread Factory (Управляемая фабрика по производству резьбы).

Подсистема EE ищет экземпляры по умолчанию в JNDI, используя имена в конфигурации привязок по умолчанию, прежде чем поместить их в стандартные имена JNDI, такие как «java:comp/DefaultManagedExecutorService»:

```
 <default-bindings
context-service="java:jboss/ee/concurrency/context/default"
datasource="java:jboss/datasources/ExampleDS"
jms-connection-factory="java:jboss/DefaultJMSConnectionFactory"
managed-executor-service="java:jboss/ee/concurrency/executor/default"
managed-scheduled-executor-
service="java:jboss/ee/concurrency/scheduler/default"
managed-thread-factory="java:jboss/ee/concurrency/factory/default" />
```

Указанные выше привязки становятся зависимостями приложения при развертывании. Однако в некоторых случаях они могут не требоваться или покрываться ресурсами, не используемыми по умолчанию. В таком случае привязка по умолчанию может быть:

– перезаписать - чтобы указать на настроенный пользователем ресурс (:write-attribute(name=…,value=…));

– не определено - если нет необходимости в зависимости от времени выполнения (:undefine-attribute(name=…)).

**Внимание:** привязки по умолчанию являются необязательными, если имя jndi для привязки по умолчанию не настроено, то соответствующий ресурс будет недоступен для приложений Jakarta EE.

**Внимание:** если ресурсы EE по умолчанию не требуются и привязки не указывают на них, можно безопасно удалить или отключить службы по умолчанию.

### 7.2 Конфигурация подсистемы присвоения имен

Подсистема именования обеспечивает реализацию JNDI в WildBoss Pro, а ее конфигурация позволяет:

– привязка записей в глобальных пространствах имен JNDI;

– выключите/включите удаленный интерфейс JNDI.

Название подсистемы - наименование, и этот документ описывает подсистему наименования версии 2.0, которая Пространство имен XML в конфигурациях WildBoss Pro XML - это «`urn:jboss:domain:naming:2.0`». Путь к XML-схеме подсистемы в дистрибутиве WildBoss Pro - это «`docs/schema/jboss-as-naming_2_0.xsd`».

Пример конфигурации подсистемы в формате XML со всеми указанными элементами и атрибутами:

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
  <bindings>
    <simple name="java:global/a" value="100" type="int" />
    <simple name="java:global/jboss.org/docs/url"
    value="https://docs.jboss.org"
    type="java.net.URL" />
    <object-factory name="java:global/foo/bar/factory"
    module="org.foo.bar" class
    ="org.foo.bar.ObjectFactory" />
    <external-context name="java:global/federation/ldap/example"
    class=
    "javax.naming.directory.InitialDirContext" cache="true">
      <environment>
        <property name="java.naming.factory.initial" value=
        "com.sun.jndi.ldap.LdapCtxFactory" />
        <property name="java.naming.provider.url" value=
        "ldap://ldap.example.com:389" />
        <property name="java.naming.security.authentication"
        value="simple" />
        <property name="java.naming.security.principal" value=
        "uid=admin,ou=system" />
        <property name="java.naming.security.credentials"
        value="secret" />
      </environment>
    </external-context>
    <lookup name="java:global/c" lookup="java:global/b" />
  </bindings>
  <remote-naming/>
</subsystem>
```

#### 7.2.1 Конфигурация глобальных привязок

Конфигурация подсистемы именования позволяет привязывать записи к следующим глобальным пространствам имен JNDI:

– «`java:global`»;

– «`java:jboss`»;

– «`java:`».

**Внимание:** если WildBoss Pro будет использоваться в качестве сервера приложений Jakarta EE, то рекомендуется выбрать «`java:global`», поскольку это стандартное (т.е. переносимое) пространство имен.

Поддерживаются четыре различных типа привязок:

– Simple;

– Object Factory;

– External Context;

– Lookup.

В конфигурации XML подсистемы глобальные привязки настраиваются с помощью элемента <bindings /> XML, например:

```
<bindings>
   <simple name="java:global/a" value="100" type="int" />
   <object-factory name="java:global/foo/bar/factory"
   module="org.foo.bar" class=
   "org.foo.bar.ObjectFactory" />
   <external-context name="java:global/federation/ldap/example" class=
   "javax.naming.directory.InitialDirContext" cache="true">
       <environment>
         <property name="java.naming.factory.initial" value=
         "com.sun.jndi.ldap.LdapCtxFactory" />
         <property name="java.naming.provider.url" value=
         "ldap://ldap.example.com:389" />
         <property name="java.naming.security.authentication"
         value="simple" />
         <property name="java.naming.security.principal" value="
         uid=admin,ou=system" />
         <property name="java.naming.security.credentials"
         value="secret" />
       </environment>
   </external-context>
   <lookup name="java:global/c" lookup="java:global/b" />
</bindings>
```

#### 7.2.1.1 Простые привязки

Простая привязка - это элементарная запись или «java.net.URL», и она определяется с помощью простого XML элемента. Пример его XML-конфигурации:

```
<simple name="java:global/a" value="100" type="int" />
```

Атрибут «name» является обязательным и указывает целевое имя JNDI для записи.

Атрибут «value» является обязательным и определяет значение записи.

Необязательный атрибут «type», который по умолчанию имеет значение «java.lang.String», задающая тип значения записи . Помимо «java.lang.String», разрешенные типы - это все примитивные типы и соответствующие им классы-оболочки объектов, такие как «int» или «java.lang.Integer» и «java.net.URL».

Управляющие клиенты, такие как WildBoss Pro CLI, могут использоваться для настройки простых привязок. Пример добавления и удаления привязок приведен в XML-примере выше:

```
/subsystem=naming/binding=java\:global\/a:add(binding-type=simple,
      type=int, value=100)
/subsystem=naming/binding=java\:global\/a:remove
```

#### 7.2.1.2 Фабрики объектов

Конфигурация подсистемы именования позволяет привязывать записи javax.naming.spi.ObjectFactory, например, с помощью элемента object-factory XML:

```
<object-factory name="java:global/foo/bar/factory"
module="org.foo.bar" class=
"org.foo.bar.ObjectFactory">
 <environment>
   <property name="p1" value="v1" />
   <property name="p2" value="v2" />
 </environment>
</object-factory>
```

Атрибут «name» является обязательным и указывает целевое имя JNDI для записи.

Атрибут «class» является обязательным и определяет Java-тип фабрики объектов.

Атрибут «module» является обязательным и указывает идентификатор модуля JBoss , из которого может быть загружен Java-класс «object factory».

Необязательный дочерний элемент «environment» может использоваться для создания пользовательской среды для фабрики объектов.

Управляющие клиенты, такие как WildBoss Pro CLI, могут использоваться для настройки привязок фабрики объектов. Пример добавления и удаления привязок приведен в XML-примере выше:

```
/subsystem=naming/binding=java\:global\/foo\/bar\/factory:add(binding-
      type=objectfactory,module=org.foo.bar,
      class=org.foo.bar.ObjectFactory, environment=[p1=v1,p2=v2])
/subsystem=naming/binding=java\:global\/foo\/bar\/factory:remove
```

#### 7.2.1.3 Объединение внешнего контекста

Объединение внешних контекстов JNDI, таких как контекст LDAP, достигается путем добавления внешних Контекстных привязок к конфигурации глобальных привязок с помощью XML-элемента «external-context». Пример его XML-конфигурации:

```
<external-context name="java:global/federation/ldap/example" class=
"javax.naming.directory.InitialDirContext" cache="true">
  <environment>
     <property name="java.naming.factory.initial" value=
     "com.sun.jndi.ldap.LdapCtxFactory" />
     <property name="java.naming.provider.url"
     value="ldap://ldap.example.com:389"
     />
     <property name="java.naming.security.authentication"
     value="simple" />
     <property name="java.naming.security.principal"
     value="uid=admin,ou=system" />
     <property name="java.naming.security.credentials" value="secret"
     />
  </environment>
</external-context>
```

Атрибут «name» является обязательным и указывает целевое имя JNDI для записи.

Атрибут «class» является обязательным и указывает на начальный тип контекста именования Java, используемый для создания объединенного контекста. Обратите внимание, что такой тип должен иметь конструктор с единственным аргументом «environment map».

Необязательный атрибут «module» указывает идентификатор модуля JBoss, из которого могут быть загружены любые классы, требуемые внешним контекстом JNDI.

Необязательный атрибут «cache», значение которого по умолчанию равно «false», указывает, следует ли кэшировать экземпляр внешнего контекста.

Необязательный дочерний элемент «environment» может использоваться для предоставления пользовательской среды, необходимой для поиска внешнего контекста.

Управляющие клиенты, такие как WildBoss Pro CLI, могут использоваться для настройки привязок внешнего контекста. Пример добавления и удаления приведен в XML-примере:

```
/subsystem=naming/binding=java\:global\/federation\/ldap\/example:add(bind
      ingtype=external-context, cache=true,
      class=javax.naming.directory.InitialDirContext,environment=[java.na
      ming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory,
java.naming.provider.url=ldap\:\/\/ldap.example.com\:389,
java.naming.security.authentication=simple,
java.naming.security.principal=uid\=admin\,ou\=system,
java.naming.security.credentials= secret])
/subsystem=naming/binding=java\:global\/federation\/ldap\/example:remove
```

У некоторых поставщиков JNDI может произойти сбой при поиске по их ресурсам, если они неправильно реализуют метод lookup(Name). Их ошибки могут выглядеть следующим образом:

```
11:31:49,047 ERROR
org.jboss.resource.adapter.jms.inflow.JmsActivation (default-
threads -1)
javax.naming.InvalidNameException: Only support CompoundName
namesat
com.tibco.tibjms.naming.TibjmsContext.lookup(TibjmsContext.java:50
4)at
javax.naming.InitialContext.lookup(InitialContext.java:421)
```

Чтобы обойти их недостатки, можно указать свойство «org.jboss.as.naming.lookup.by. string» в среде внешнего контекста, чтобы вместо него использовать метод lookup(String) (что приведет к снижению производительности):

```
<property name="org.jboss.as.naming.lookup.by.string" value="true"/>
```

#### 7.2.1.4 Обязательные псевдонимы

Конфигурация подсистемы именования позволяет привязывать существующие записи к дополнительным именам, т.е. псевдонимам. Псевдонимы привязки задаются с помощью элемента «lookup» XML. Пример ее конфигурации в формате XML:

```
<lookup name="java:global/c" lookup="java:global/b" />
```

Атрибут «name» является обязательным и указывает целевое имя JNDI для записи.

Атрибут «lookup» является обязательным и указывает исходное имя JNDI. Он может объединять поисковые запросы во внешних контекстах. Например, при наличии внешнего контекста, связанного с «java:global/federation/ldap/example», поиск можно выполнить там, установив атрибут поиска в «java:global/federation/ldap/example/subfolder».

Управляющие клиенты, такие как WildBoss Pro CLI, могут использоваться для настройки псевдонимов привязки. Пример добавления и удаления псевдонимов приведен в XML-примере выше:

```
/subsystem=naming/binding=java\:global\/c:add(binding-type=lookup,
lookup=java\:global\/b)
/subsystem=naming/binding=java\:global\/c:remove
```

### 7.2.2 Удаленная настройка JNDI

Конфигурация подсистемы именования может использоваться для (де)активации удаленного интерфейса JNDI, который позволяет клиентам просматривать записи, присутствующие в удаленном экземпляре WildBoss Pro.

Внимание: только записи в контексте «java:jboss/exported» доступны через удаленный JNDI.

В конфигурации XML подсистемы привязки удаленного доступа JNDI настраиваются с помощью элемента XML <удаленное присвоение имен />:

```
<remote-naming />
```

Управляющие клиенты, такие как WildBoss Pro CLI, могут использоваться для добавления/удаления удаленного интерфейса JNDI. Пример добавления и удаления интерфейса приведен в XML-примере выше:

```
/subsystem=naming/service=remote-naming:add
/subsystem=naming/service=remote-naming:remove
```

## 7.3 Конфигурация источника данных

Источники данных настраиваются через подсистему «datasource». Объявление нового источника данных состоит из двух отдельных шагов: вам нужно будет предоставить драйвер JDBC и определить источник данных, который ссылается на установленный вами драйвер.

### 7.3.1 Установка драйвера JDBC

Рекомендуемый способ установки драйвера JDBC в WildBoss Pro 26.1 – это развернуть его как обычное JAR -развертывание. Причина этого заключается в том, что при запуске WildBoss Pro в режиме домена, развертывания автоматически распространяются на все серверы, к которым применяется развертывание; таким образом, вам не нужно беспокоиться о распространении JAR-файла драйвера!

Любой драйвер, совместимый с JDBC 4, будет автоматически распознан и установлен в систему по имени и версии. JAR-файл JDBC идентифицируется с помощью механизма поставщика услуг Java. Такие JAR-файлы будут содержать текстовый файл с именем «`META-INF/services/java.sql.Driver`», который содержит названия классов драйверов, существующих в этом JAR-файле. Если ваш JAR-файл драйвера JDBC не совместим с JDBC 4, его можно сделать доступным для развертывания одним из нескольких способов.

#### 7.3.1.1 Измените банку

Самое простое решение – просто изменить JAR и добавить отсутствующий файл. Вы можете сделать это из своей командной оболочки, выполнив:

– измените или создайте пустой временный каталог;

– создайте подкаталог «META-INF»;

– создайте подкаталог «META-INF/services»;

– создайте файл «META-INF/services/java.sql.Driver», содержащий одну строку - полное имя класса драйвера JDBC;

– Используйте инструмент командной строки jar для обновления JAR следующим образом:

```
/subsystem=naming/service=remote-naming:add
/subsystem=naming/service=remote-naming:remove
```

Подробное объяснение того, как развернуть jar-драйвер, совместимый с JDBC 4, приведено в главе «Развертывание приложения».

### 7.3.2 Определения источников данных

Сам источник данных определяется в подсистеме datasources:

```
<subsystem xmlns="urn:jboss:domain:datasources:4.0">
    <datasources>
        <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-
        name="ExampleDS">
            <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
            <driver>h2</driver>
            <pool>
                <min-pool-size>10</min-pool-size>
                <max-pool-size>20</max-pool-size>
                <prefill>true</prefill>
            </pool>
            <security>
                <user-name>sa</user-name>
                <password>sa</password>
            </security>
        </datasource>
        <xa-datasource jndi-name="java:jboss/datasources/ExampleXADS" pool-name=
        "ExampleXADS">
        <driver>h2</driver>
        <xa-datasource-property name="URL">jdbc:h2:mem:test</xa-datasourceproperty>
            <xa-pool>
                <min-pool-size>10</min-pool-size>
                <max-pool-size>20</max-pool-size>
                <prefill>true</prefill>
            </xa-pool>
            <security>
                <user-name>sa</user-name>
                <password>sa</password>
            </security>
            </xa-datasource>
            <drivers>
            <driver name="h2" module="com.h2database.h2">
        <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
        </driver>
        </drivers>
    </datasources>
</subsystem>
```

(Смотри «`standalone/configuration/standalone.xml`»)

Как вы можете видеть, источник данных ссылается на драйвер по его логическому имени.

Вы можете легко запросить ту же информацию через интерфейс командной строки:

```
[standalone@localhost:9990 /] /subsystem=datasources:read-
      resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "data-source" => {"H2DS" => {
    "connection-url" => "jdbc:h2:mem:test;DB_CLOSE_DELAY=-1",
    "jndi-name" => "java:/H2DS",
    "driver-name" => "h2",
    "pool-name" => "H2DS",
    "use-java-context" => true,
    "enabled" => true,
    "jta" => true,
    "pool-prefill" => true,
    "pool-use-strict-min" => false,
    "user-name" => "sa",
    "password" => "sa",
    "flush-strategy" => "FailingConnectionOnly",
    "background-validation" => false,
    "use-fast-fail" => false,
    "validate-on-match" => false,
    "use-ccm" => true
  }},
  "xa-data-source" => undefined,
  "jdbc-driver" => {"h2" => {
    "driver-name" => "h2",
    "driver-module-name" => "com.h2database.h2",
    "driver-xa-datasource-class-name" => "org.h2.jdbcx.JdbcDataSource"
  }}
 }
}
[standalone@localhost:9990 /] /subsystem=datasources:installed-drivers-
      list
{
  "outcome" => "success",
  "result" => [{
    "driver-name" => "h2",
    "datasource-class-info" => [{"org.h2.jdbcx.JdbcDataSource" => {
      "URL" => "java.lang.String",
      "description" => "java.lang.String",
      "loginTimeout" => "int",
      "password" => "java.lang.String",
      "url" => "java.lang.String",
      "user" => "java.lang.String"
    }}],
    "deployment-name" => undefined,
    "driver-module-name" => "com.h2database.h2",
    "module-slot" => "main",
    "driver-xa-datasource-class-name" =>
    "org.h2.jdbcx.JdbcDataSource",
    "driver-class-name" => "org.h2.Driver",
    "driver-major-version" => 1,
```

```
    "driver-minor-version" => 3,
    "jdbc-compliant" => true
  }]
}
```

**Внимание:** «datasource-class-info» отображает свойства соединения, определенные в «(xa-)datasource-class».

**Внимание:** использование веб-консоли или интерфейса командной строки значительно упрощает развертывание драйверов JDBC и создание источников данных.

Интерфейс командной строки предлагает набор команд для создания и изменения источников данных:

```
[standalone@localhost:9990 /] data-source --help
SYNOPSIS
  data-source --help [--properties | --commands] |
    (--name=<resource_id> (--<property>=<value>)*) |
    (<command> --name=<resource_id> (--<parameter>=<value>)*)
    [--headers={<operation_header> (;<operation_header>)*}]
DESCRIPTION
 The command is used to manage resources of type
 /subsystem=datasources/data-source.
[...]
[standalone@localhost:9990 /] xa-data-source --help
SYNOPSIS
  xa-data-source --help [--properties | --commands] |
     (--name=<resource_id> (--<property>=<value>)*) |
     (<command> --name=<resource_id> (--<parameter>=<value>)*)
     [--headers={<operation_header> (;<operation_header>)*}]
DESCRIPTION
 The command is used to manage resources of type
 /subsystem=datasources/xa-datasource.
RESOURCE DESCRIPTION
 A JDBC XA data-source configuration
[...]
```

### 7.3.3 Ссылка на компонент

Подсистема «datasource» предоставляется проектом IronJacamar. Для получения подробного описания доступных свойств конфигурации, пожалуйста, обратитесь к документации проекта.

– Домашняя страница IronJacamar: http://ironjacamar.org/.

– Проектная документация: http://ironjacamar.org/documentation.html.

– Описание схемы: http://www.ironjacamar.org/doc/userguide/1.1/en-US/html_single/index.html#развертываниеds_descriptor.

## 7.4 Конфигурация Agroal

Подсистема Agroal позволяет определять источники данных. Объявление нового источника данных состоит из двух отдельных шагов: предоставления драйвера JDBC и определения источника данных, который ссылается на установленный вами драйвер.

Подсистема Agroal предоставляется проектом Agroal. Для получения подробного описания доступных свойств конфигурации, пожалуйста, ознакомьтесь с проектной документацией.

### 7.4.1 Включение подсистемы

Если в конфигурации WildBoss Pro подсистема Agroal не включена по умолчанию, ее можно включить следующими способами.

```
<extensions>
   <extension module="org.WildBoss Pro.extension.datasources-agroal"/>
   [...]
</extensions>
<subsystem xmlns="urn:jboss:domain:agroal:1.0">
   [...]
</subsystem>
```

```
[standalone@localhost:9990 /] /extension=org.WildBoss
      Pro.extension.datasources-agroal:add
{"outcome" => "success"}
[standalone@localhost:9990 /] /subsystem=datasources-agroal:add
{
   "outcome" => "success",
   "response-headers" => {
   "operation-requires-reload" => true,
   "process-state" => "reload-required"
   }
}
```

### 7.4.2 Установка драйвера JDBC

Определение драйвера - это ссылка на класс в драйвере JDBC. В одном и том же драйвере JDBC можно создать несколько определений для нескольких классов в нем. Для Agroal требуется реализация «`java.sql.Driver`» или «`javax.sql.DataSource`» для источников данных, отличных от XA, в то время как для XA требуется реализация «`javax.sql.XADataSource`».

**Внимание:** Agroal попытается загрузить «`java.sql.Driver`» из указанного модуля, если класс не определен.

**Внимание:** любой установленный драйвер предоставляет операцию под названием «`class-info`», в которой перечислены все свойства, доступные для этого конкретного класса, которые могут быть установлены в «`connectionfactory`».

```
<subsystem xmlns="urn:jboss:domain:agroal:1.0">
   [...]
   <drivers>
      <driver name="h2" module="com.h2database.h2"
      class="org.h2.Driver"/>
   </drivers>
</subsystem>
```

```
[standalone@localhost:9990 /] /subsystem=datasources-
      agroal/driver=h2:read-resource
{
  "outcome" => "success",
  "result" => {
  "class" => "org.h2.Driver",
  "module" => "com.h2database.h2"
  }
}
```

### 7.4.3 Общие определения источников данных

Agroal предоставляет как XA-, так и не-XA-источники данных, и большинство атрибутов, которые их определяют, являются общими. Это определение в основном разделено на две логические единицы: фабрику подключений и пул подключений. Как следует из названия, фабрика подключений содержит все, что требуется для создания новых подключений, а пул подключений определяет, как пул обрабатывает соединения.

#### 7.4.3.1 Заводское определение подключения

Для установки на заводе-изготовителе требуется ссылка на драйвер (см. «agroal-driver-installation»). С помощью «java.sql.Driver» предпочтительным способом «указать» на базу данных является указание атрибута «url», в то время как для «javax.sql.DataSource» и «javax.sql.XADataSource» предпочтительным способом является указание свойств подключения.

**Внимание:** атрибуты «username» и «password» предоставляются для базовой аутентификации в базе данных. Agroal не позволяет устанавливать имя пользователя и пароль в качестве свойств подключения из- за требований безопасности.

Другие возможности, предоставляемые определением «connection-factory», включают возможность выполнения инструкции SQL сразу после создания соединения и указания уровня изоляции транзакций в базе данных.

```
<subsystem xmlns="urn:jboss:domain:agroal:1.0">
  <datasource [...]>
  [...]
     <connection-factory driver="h2"
     url="jdbc:h2:tcp://localhost:1701"
     transaction-isolation="SERIALIZABLE" new-connection-sql="SELECT
     1" username="sa"
     password="sa">
        <connection-properties>
           <property name="aProperty" value="aValue"/>
           <property name="anotherProperty" value="anotherValue"/>
        </connection-properties>
     </connection-factory>
  </datasource>
  [...]
</subsystem>
```

```
---
[standalone@localhost:9990 /] /subsystem=datasources-
      agroal/datasource=sample:readresource
{
  "outcome" => "success",
  "result" => {
    "connection-factory" => {
      "driver" => "h2",
      "url" => "jdbc:h2:tcp://localhost:1701",
      "transaction-isolation" => "SERIALIZABLE",
      "new-connection-sql" => "SELECT 1",
      "username" => "sa",
      "password" => "sa",
      "connection-properties" => {
         "aProperty" => "aValue",
         "anotherProperty" => "anotherValue"
      }
    }
    [...]
  }
}
---
```

#### 7.4.3.2 Общие атрибуты источника данных

Все источники данных в Agroal имеют имя, которое используется для их определения в модели среды выполнения WildBoss Pro, и привязаны к имени JNDI.

Атрибут «statistics-enabled» позволяет собирать показатели, относящиеся к пулу, которые могут быть запрошены в модели среды выполнения.

**Внимание:** также предусмотрена операция сброса статистики.

```
<subsystem xmlns="urn:jboss:domain:agroal:1.0">
  <xa-datasource name="sample-xa" jndi-
  name="java:jboss/datasources/ExampleXADS"
statistics-enabled="true">
      [...]
  </xa-datasource>
  [...]
</subsystem>
```

```
---
[standalone@localhost:9990 /] /subsystem=datasources-
      agroal/datasource=samplexa:read-resource
{
   "outcome" => "success",
   "result" => {
     "jndi-name" => "java:jboss/datasources/ExampleXADS",
     "statistics-enabled" => true
     [...]
   }
}
---
```

Доступная статистика включает количество созданных/уничтоженных подключений и количество подключений, используемых/доступных в пуле. Также доступна статистика по

времени, которое требуется для создания соединения, и по тому, как долго потоки были заблокированы в ожидании соединения.

```
[standalone@localhost:9990 /] /subsystem=datasources-
      agroal/datasource=sample:readresource(
include-runtime)
{
  "outcome" => "success",
  "result" => {
    "statistics" => {
       "acquire-count" => 10L,
       "active-count" => 3L,
       "available-count" => 17L,
       "awaiting-count" => 0L,
       "creation-count" => 20L,
       "destroy-count" => 0L,
       "flush-count" => 0L,
       "invalid-count" => 0L,
       "leak-detection-count" => 0L,
       "max-used-count" => 20L,
       "reap-count" => 0L,
       "blocking-time-average-ms" => 0L,
       "blocking-time-max-ms" => 0L,
       "blocking-time-total-ms" => 0L,
       "creation-time-average-ms" => 96L,
       "creation-time-max-ms" => 815L,
       "creation-time-total-ms" => 964L
    }
    [...]
  }
}
```

#### 7.4.3.3 Атрибуты, специфичные для источника данных

Помимо всех общих атрибутов, определение источника данных может отключить Jakarta Transactions интеграция.

Отложенная регистрация не поддерживается, если включена функция Jakarta Transactions, соединение всегда должно быть получено в рамках транзакции. Соединение всегда будет зарегистрировано с помощью этой транзакции (отложенная регистрация не поддерживается).

**Внимание:** атрибут «`connectable`» позволяет источнику данных, отличному от XA, принимать участие в транзакции XA ('Last Resource Commit Optimization (LRCO)' / 'Commit Markable Resource').

```
<subsystem xmlns="urn:jboss:domain:agroal:1.0">
 <datasource name="sample" jndi-
 name="java:jboss/datasources/ExampleDS" jta="false"
 connectable="false" statistics-enabled="true">
   [...]
 </datasource>
 [...]
</subsystem>
```

```
---
[standalone@localhost:9990 /] /subsystem=datasources-
      agroal/datasource=samplexa:read-resource
{
  "outcome" => "success",
  "result" => {
    "connectable" => false,
    "jta" => false,
    [...]
  }
}
---
```

#### 7.4.3.4 Специфичные атрибуты XADataSource

На данный момент нет атрибутов, специфичных для определения XADataSource.

## 7.5 Конфигурация ведения журнала

Общая конфигурация ведения журнала сервера представлена подсистемой ведения журнала. Она состоит из четырех основных частей: конфигурации обработчиков, средство ведения журнала, объявления корневого средства ведения журнала (они же категории журналов) и профили ведения журнала. Каждое средство ведения журнала ссылается на обработчик (или набор обработчиков). Каждый обработчик объявляет формат журнала и выходные данные:

```
<subsystem xmlns="urn:jboss:domain:logging:3.0">
   <console-handler name="CONSOLE" autoflush="true">
      <level name="DEBUG"/>
      <formatter>
          <named-formatter name="COLOR-PATTERN"/>
      </formatter>
   </console-handler>
   <periodic-rotating-file-handler name="FILE" autoflush="true">
      <formatter>
          <named-formatter name="PATTERN"/>
      </formatter>
      <file relative-to="jboss.server.log.dir" path="server.log"/>
      <suffix value=".yyyy-MM-dd"/>
   </periodic-rotating-file-handler>
   <logger category="com.arjuna">
      <level name="WARN"/>
   </logger>
   [...]
   <root-logger>
      <level name="DEBUG"/>
      <handlers>
```

```
            <handler name="CONSOLE"/>
            <handler name="FILE"/>
        </handlers>
    </root-logger>
    <formatter name="PATTERN">
        <pattern-formatter pattern="%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t)
        %s%e%n"/>
    </formatter>
    <formatter name="COLOR-PATTERN">
        <pattern-formatter pattern="%K{level}%d{HH:mm:ss,SSS} %-5p [%c] (%t)
        %s%e%n"/>
    </formatter>
</subsystem>
```

### 7.5.1 Атрибуты

Корневой ресурс содержит два примечательных атрибута: «`add-logging-api-dependencies`» и «`usedeployment- logging-config`».

***add-logging-api-dependencies***

Параметр «`add-logging-api-dependencies`» определяет, добавляет ли контейнер неявные зависимости API ведения журнала в ваши развертывания. Если установлено значение «`true`», то по умолчанию добавляются все неявные зависимости API ведения журнала. Если установлено значение «`false`», то зависимости не добавляются в ваши развертывания.

***use-deployment-logging-config***

Параметр «`use-deployment-logging-config`» определяет, проверяется ли ваше развертывание на предмет ведения журнала для каждого развертывания. Если установлено значение «`true`», то по умолчанию ведение журнала для каждого развертывания включено. Установите значение «`false`», чтобы отключить эту функцию.

### 7.5.2 Ведение журнала для каждого развертывания

Ведение журнала для каждого развертывания позволяет вам добавить файл конфигурации ведения журнала в ваше развертывание и настроить ведение журнала для этого развертывания в соответствии с файлом конфигурации. В EAR конфигурация должна находиться в каталоге META-INF. При развертывании WAR или JAR файл конфигурации может находиться либо в каталогах META-INF, либо в каталогах WEB-INF/classes.

Разрешены следующие конфигурационные файлы:

– logging.properties;
– jboss-logging.properties;
– log4j.properties;
– log4j.xml;
– jboss-log4j.xml.

Вы также можете отключить эту функцию, изменив значение атрибута «`use-deployment-logging-config`» на «`false`».

### 7.5.3 Профили ведения журнала

Профили ведения журнала подобны дополнительным подсистемам ведения журнала. Каждый профиль ведения журнала состоит из трех из четырех основных частей, перечисленных выше: конфигурации обработчика, регистратора и объявления корневого регистратора.

Вы можете назначить профиль ведения журнала развертыванию с помощью манифеста развертываний. Добавьте запись в журнал- Запись профиля в файл «MANIFEST.MF» со значением идентификатора профиля ведения журнала. Например, профиль ведения журнала, определенный в «`/subsystem=logging/logging-profile=ejbs`» MANIFEST.MF будет выглядеть следующим образом:

```
Manifest-Version: 1.0
Logging-Profile: ejbs
```

Профиль ведения журнала может быть назначен любому количеству развертываний. Использование профиля ведения журнала также позволяет вносить изменения в конфигурацию во время выполнения. Это преимущество перед конфигурацией ведения журнала для каждого развертывания, поскольку для внесения изменений в журнал не требуется повторное развертывание.

### 7.5.4 Расположения файлов журнала по умолчанию

#### 7.5.4.1 Управляемый домен

В управляемом домене существуют два типа файлов журналов: журналы контроллера и журналы сервера. Компоненты контроллера управляют доменом в целом. В их обязанности входит запуск/остановка экземпляров сервера и выполнение управляемых операций по всему домену. Журналы сервера содержат информацию для регистрации конкретного экземпляра сервера. Они расположены рядом с хостом, на котором запущен сервер.

Для простоты мы рассмотрим настройки по умолчанию для управляемого домена. В этом случае компоненты контроллера домена и серверы расположены на одном хосте:

| Процесс | Файл журнала |
|---|---|
| Главный контроллер (Host Controller) | ./domain/log/host-controller.log |
| Контроллер процесса (Process Controller) | ./domain/log/process-controller.log |
| «Первый сервер» ("Server One") | ./domain/servers/server-one/log/server.log |
| «Второй сервер» ("Server Two") | ./domain/servers/server-two/log/server.log |
| «Третий сервер» ("Server Three") | ./domain/servers/server-three/log/server.log |

#### 7.5.4.2 Автономный сервер

Файлы журнала по умолчанию для автономного сервера можно найти в подкаталоге журнала дистрибутива:

| Процесс | Файл журнала |
|---|---|
| Сервер (Server) | ./standalone/log/server.log |

### 7.5.5 Список файлов журналов и чтение файлов журналов

Файлы журналов могут быть перечислены и просмотрены с помощью операций управления. Разрешенные к просмотру файлы журналов намеренно ограничены файлами, которые существуют в каталоге «jboss.server.log.dir» и связаны с известным обработчиком файлов. Известные типы обработчиков файлов включают в себя обработчик файлов, обработчик файлов с периодической ротацией и обработчик файлов с ротацией размера. Операции допустимы как в автономном режиме, так и в режиме домена.

#### 7.5.5.1 Список файлов журнала

Подсистема ведения журнала имеет ресурс файла журнала вне корневого ресурса подсистемы и каждого ресурса профиля ведения журнала для отображения каждого файла журнала.

*Команда CLI и вывод данных*

```
[standalone@localhost:9990 /] /subsystem=logging:read-children-
      names(child-type=logfile)
{
  "outcome" => "success",
  "result" => [
  "server.log",
  "server.log.2014-02-12",
  "server.log.2014-02-13"
  ]
}
```

#### 7.5.5.2 Чтение файла журнала

Операция чтения файла журнала доступна для каждого ресурса файла журнала. У этой операции есть 4 необязательных параметра.

| Параметр | Описание |
|---|---|
| кодировка (encoding) | кодировка, в которой файл должен быть прочитан |
| линии (lines) | количество строк в файле. Значение «-1» указывает на то, что все строки должны быть прочитаны |
| пропускать (skip) | количество строк, которые нужно пропустить перед чтением |
| хвост (tail) | значение «true» для чтения с конца файла вверх или значение «false» для чтения сверху вниз |

*Команда CLI и вывод данных*

```
[standalone@localhost:9990 /] /subsystem=logging/log-
      file=server.log:read-log-file
{
  "outcome" => "success",
  "result" => [
    "2014-02-14 14:16:48,781 INFO
[org.jboss.as.server.deployment.scanner] (MSC service thread 1-11)
JBAS015012: Started FileSystemDeploymentService for directory
/home/jperkins/servers/WildBoss Pro-
8.0.0.Final/standalone/deployments",
```

```
    "2014-02-14 14:16:48,782 INFO
[org.jboss.as.connector.subsystems.datasources] (MSC service thread 1-
8) JBAS010400: Bound data source [java:jboss/myDs]",
    "2014-02-14 14:16:48,782 INFO
[org.jboss.as.connector.subsystems.datasources] (MSC service thread 1-
15) JBAS010400: Bound data source [java:jboss/datasources/ExampleDS]",
    "2014-02-14 14:16:48,786 INFO [org.jboss.as.server.deployment]
(MSC service thread 1-9) JBAS015876: Starting deployment of \"simple-
servlet.war\" (runtime-name: \"simple-servlet.war\")",
    "2014-02-14 14:16:48,978 INFO [org.jboss.ws.common.management]
(MSC service thread 1-10) JBWS022052: Starting JBoss Web Services -
Stack CXF Server 4.2.3.Final",
    "2014-02-14 14:16:49,160 INFO [org.WildBoss
Pro.extension.undertow] (MSC service thread 1-16) JBAS017534:
Registered web context: /simple-servlet",
    "2014-02-14 14:16:49,189 INFO [org.jboss.as.server] (Controller
Boot Thread) JBAS018559: Deployed \"simple-servlet.war\" (runtime-name
: \"simple-servlet.war\")",
    "2014-02-14 14:16:49,224 INFO [org.jboss.as] (Controller Boot
Thread) JBAS015961: Http management interface listening on
http://127.0.0.1:9990/management",
    "2014-02-14 14:16:49,224 INFO [org.jboss.as] (Controller Boot
Thread) JBAS015951: Admin console listening on http://127.0.0.1:9990",
    "2014-02-14 14:16:49,225 INFO [org.jboss.as] (Controller Boot
Thread) JBAS015874: WildBoss Pro {WildBoss ProVersion}.0.0.Final
\"WildBoss Pro\" started in 1906ms - Started 258 of 312 services (90
services are lazy, passive or on-demand)"
  ]
}
```

### 7.5.6 Часто задаваемые вопросы

Почему существует файл «logging.properties»?

Возможно, вы заметили, что в каталоге конфигурации есть файл «logging.properties». Эта конфигурация ведения журнала используется при загрузке сервера до тех пор, пока не включится подсистема ведения журнала. Если подсистема ведения журнала не включена в вашу конфигурацию, то она будет действовать как конфигурация ведения журнала для всего сервера.

Внимание: файл «logging.properties» перезаписывается при загрузке и при каждом изменении подсистемы ведения журнала. Любые изменения, внесенные в файл, не сохраняются. Любые изменения, внесенные в конфигурацию XML или с помощью операций управления, будут сохранены в файле «logging.properties» и использованы при следующей загрузке.

### 7.5.7 Форматировщики протоколирования

Для форматирования сообщения журнала используются средства форматирования. Средство форматирования может быть назначено обработчику ведения журнала.

Подсистема ведения журнала включает в себя 4 типа обработчиков:

– средство форматирования JSON (JSON Formatter);

– средство формирования шаблонов (Pattern Formatter);

– средство форматирования XML (XML Formatter);

– пользовательский форматировщик (Custom Formatter).

#### 7.5.7.1 Средство форматирования JSON

Средство форматирования, используемое для форматирования сообщений журнала в формате JSON.

***Примеры:***

*Простой форматировщик JSON*

```
/subsystem=logging/json-formatter=json:add(pretty-print=true,
exception-outputtype=formatted)
```

*Средство форматирования Logstash*

```
 /subsystem=logging/json-formatter=logstash:add(exception-output-
 type=formatted, keyoverrides=[timestamp="@timestamp"],
 meta-data=[@version=1])
```

#### 7.5.7.2 Форматировщик шаблонов

Средство форматирования, используемое для форматирования сообщений журнала в виде обычного текста. В следующей таблице описаны символы формата для средства форматирования шаблонов (таблица 4).

**Внимание:** выделенные символы указывают на то, что требуется вычисление вызывающего абонента, решение которого может оказаться дорогостоящим.

Таблица 4 - Синтаксис шаблона

| Символ | Описание | Примеры |
|---|---|---|
| %c | Категория события, регистрируемого в журнале. Для изменения категории, разделенной точками, можно использовать спецификатор точности | %c<br>org.jboss.example.Foo<br>%c{1} Foo<br>%c{2}<br>example.Foo<br>%c{.} ...Foo<br>%c{1.} o.j.e.Foo<br>%c{1~.} o.~.~.Foo |
| %C | Класс кода, вызывающего метод «log». Для изменения имени класса, разделенного точками, можно использовать спецификатор точности | %C<br>org.jboss.example.Foo<br>%C{1} Foo<br>%C{2}<br>example.Foo<br>%C{.} ...Foo<br>%C{1.} o.j.e.Foo<br>%C{1~.} o.~.~.Foo |
| %d | Временная метка сообщения журнала. Любой допустимый шаблон «SimpleDateFormat». Значение по умолчанию - гггг-ММ-дд ЧЧ:мм:сс,SSS | %d{HH:mm:ss,SSS}<br>%d{yyyy-MMdd'T'HH:<br>mm:ss.SSS<br>XXX} |

| Символ | Описание | Примеры |
|---|---|---|
| %D | Имя модуля, из которого пришло сообщение журнала. Для изменения имени модуля, разделенного точками, можно использовать спецификатор точности | %D<br>org.jboss.example<br>%D{1} example<br>%D{2}<br>jboss.example<br>%D{.} ..example<br>%D{1.}<br>o.j.example<br>%D{1~.}<br>o.~.example |
| %e | Трассировка стека исключений. Принимает аргумент, указывающий, сколько уровней подавленных сообщений необходимо напечатать | %e Prints the full<br>stack trace.<br>%e Prints the stack<br>{0 trace ignoring any<br>} suppressed<br>messages.<br>%e Prints the stack<br>{1 trace with a<br>} maximum of one<br>suppressed<br>message |
| %F | Имя файла класс, который зарегистрировал сообщение | |
| %h | Краткое имя хоста. Это будет первая часть полного имени хоста | %h localhost |
| %H | Полное имя хоста. Для изменения имени хоста, разделенного точками, можно использовать спецификатор точности | %H<br>developer.jboss.o<br>rg<br>%H{1} developer |
| %i | Идентификатор процесса | |
| %k | Идентификатор процесса | |
| %K | Если поддерживается цветной вывод, определяет цвета, которые будут отображаться в сообщении журнала | %K The level<br>{l determines the<br>Ev color of the<br>El output.<br>}<br>%K All messages will<br>{r be colored red<br>ed<br>} |
| %l | Информация о местоположении. Это включает в себя имя вызывающего класса, имя метода, имя файла и номер строки | %l<br>org.jboss.example.<br>Foo.bar(Foo.java:33) |
| %L | Номер линии вызывающего абонента | |
| %m | Отформатированное сообщение, включающее любые трассировки стека | |
| %M | Имя вызывающего метода | |
| %n | Независимый от платформы разделитель линий | |

| Символ | Описание | Примеры |
|---|---|---|
| %N | Название процесса | |
| %p | Уровень регистрируемого сообщения | |
| %P | Локализованный уровень зарегистрированного сообщения | |
| %r | Относительное количество миллисекунд, прошедших с заданного базового времени из сообщения журнала | |
| %s | Сообщение в простом формате. Это не будет включать трассировку стека, если причина была зарегистрирована | |
| %t | Имя потока вызывающих абонентов | |
| %v | Версия модуля. Для изменения версии модуля, разделенной точками, можно использовать спецификатор точности | |
| %x | Вложенные записи диагностического контекста. Для указания количества записей для печати можно использовать спецификатор точности | `%x` `value1.value2.val` `ue3` `%x{1} value3` `%x{2}` `value2.value3` |
| %X | Запись сопоставленного диагностического контекста. За этой записью должен следовать ключ для записи MDC | `%X{key}` |
| %z | Позволяет переопределять часовой пояс при форматировании временной метки. Это значение должно предшествовать временной метке | `%z{GMT}%d{yyyy-MMdd'T'HH:mm:ssSSSXXX}` |
| %# | Позволяет добавлять системное свойство к сообщению журнала | `%#{jboss.server.name}` |
| %$ | Позволяет добавлять системное свойство к сообщению журнала | `%${jboss.server.name}` |
| %% | Экранирует символ «%» | |

Вы также можете изменить формат, поместив необязательный модификатор формата между знаком процента и символом (таблица 5).

Таблица 5 - Примеры модификаторов формата

| Модификатор | Выравнивание по левому краю | Минимальная ширина | Максимальная ширина | Пример |
|---|---|---|---|---|
| [%20c] | false | 20 | | `[ org.jboss.example]` |
| [%-20c] | true | 20 | | `[org.jboss.example ]` |
| [%.10c] | | | 10 | `[org.jboss]` |
| [%20.30c] | false | 20 | 30 | `[ org.jboss.example]` |
| [%-20.30c] | true | 20 | 30 | `[org.jboss.example ]` |

***Примеры***

*Простой форматировщик шаблонов:*

```
/subsystem=logging/pattern-
formatter=DEFAULT:add(pattern="%d{HH:mm:ssSSSXXX} %-5p [%c](%t)
%10.10#{jboss.node.name} %s%e%n")
```

*Средство формирования цветового рисунка:*

```
/subsystem=logging/pattern-
formatter=DEFAULT:add(colormap="info:cyan,warn:brightyellow,error:brigh
tred,debug:magenta",pattern="%K{level}%d{yyyy-MM-dd'T'HH:mm:ssSSSXXX}
%-5p [%c] (%t) %s%e%n")
```

#### 7.5.7.3 Средство форматирования XML

Средство форматирования, используемое для форматирования сообщений журнала в формате XML.

***Примеры***

*Простой форматировщик XML:*

```
/subsystem=logging/xml-formatter=xml:add(pretty-print=true, exception-
outputtype=detailed-and-formatted)
```

*Ключ переопределяет средство форматирования XML:*

```
/subsystem=logging/xml-formatter=xml:add(pretty-print=true, print-
namespace=true,namespace-uri="urn:custom:1.0", key-
overrides={message=msg, record=logRecord,timestamp=date}, print-
details=true)
```

#### 7.5.7.4 Пользовательский форматировщик

Пользовательский форматировщик, который будет использоваться с обработчиками. Обратите внимание, что большинство записей журнала отформатированы в формате «printf». Для форматирования может потребоваться вызов «org.jboss.logmanager.ExtLogRecord#getFormattedMessage()» для правильного форматирования сообщения.

***Пример:***

```
/subsystem=logging/customformatter=custom:add(class=org.jboss.example.C
ustomFormatter,module=org.jboss.example,properties={prettyPrint=true,pr
intDetails=true,bufferSize=1024})
```

### 7.5.8 Обработчики

#### 7.5.8.1 Обзор

Обработчики определяют, как записываются сообщения журнала. Если средство регистрации сообщает, что сообщение доступно для регистрации, оно затем обрабатывается обработчиком журнала.

Далее по тексту приведены доступные обработчики для WildBoss-Pro:

– асинхронный обработчик (async-handler) – это обработчик, который асинхронно записывает сообщения журнала в свои дочерние обработчики. Этот тип обработчика обычно

используется для переноса других обработчиков, которым требуется значительное время для записи сообщений.;

– консоль-обработчик (console-handler) – это обработчик, который записывает сообщения журнала в консоль. Обычно он выполняет запись в стандартный вывод, но может быть настроен на запись в stderr;

– пользовательский обработчик (custom-handler) позволяет пользователю определить любой обработчик как обработчик, который может быть назначен регистратору или асинхронному обработчику;

– обработчик файлов (file-handler) – это обработчик, который записывает лог-сообщения в указанный файл;

– обработчик периодической ротации файлов (periodic-rotating-file-handler) – это обработчик, который записывает сообщения журнала в указанный файл. Файл ротируется в соответствии с датой, указанной в атрибуте «suffix». Суффикс должен быть допустимым шаблоном, распознаваемым «java.text.SimpleDateFormat», и не должен изменяться на секунды или миллисекунды.

**Внимание:** поворот происходит до того, как обработчик запишет следующее сообщение;

– обработчик периодического изменения размера файла (periodic-size-rotating-file-handler) – это обработчик, который записывает сообщения журнала в указанный файл. Файл изменяется в соответствии с датой, указанной в атрибуте «suffix» или атрибуте «rotate-size». Суффикс должен быть допустимым шаблоном, распознаваемым «java.text.SimpleDateFormat», и не должен изменяться на секунды или миллисекунды.

Параметр «max-backup-index» в этом обработчике работает иначе, чем параметр «size-rotating-file-handler». Суффикс даты файла, который необходимо изменить, должен совпадать с текущим ожидаемым суффиксом. Например, при использовании суффикса «ГГГГ-ММ» и размера при повороте на 10 м файл будет поворачиваться в соответствии с текущим месяцем каждый раз, когда будет достигнут размер в 10 Мб.

**Внимание:** поворот происходит до того, как обработчик запишет следующее сообщение;

– обработчик изменения размера файла (size-rotating-file-handler) – это обработчик, который записывает сообщения журнала в указанный файл. Файл поворачивается, когда размер файла превышает значение атрибута «rotate-size». Измененный файл будет сохранен, а к его имени будет добавлен индекс, увеличивающий индексы ранее измененных файлов на 1, пока не будет достигнут максимальный резервный индекс. Как только будет достигнут максимальный резервный индекс, проиндексированные файлы будут перезаписаны.

**Внимание:** поворот происходит до того, как обработчик запишет следующее сообщение;

– обработчик сокетов (socket-handler) – это обработчик, который отправляет сообщения через сокет. Это может быть сокет TCP или UDP и должен быть определен в группе привязки сокетов в ресурсе «local-destination-outbound-socketbinding» или «remote-destination-outbound-socketbinding».

Во время загрузки сообщения протоколирования будут помещаться в очередь до тех пор, пока не будет настроена привязка к сокету и не будет добавлена подсистема протоколирования. Это важно отметить, поскольку установка уровня обработчика равным «ОТЛАДКА» или «ТРАССИРОВКА» могут привести к большому потреблению памяти во время загрузки.

**Внимание:** сервер, загруженный в режиме только для администратора, будет отбрасывать сообщения, а не отправлять их через сокет.

*Пример интерфейса командной строки*

```
# Add the socket binding
/socket-binding-group=standard-sockets/remote-destination-outbound-
socket-binding=logserver:add(host=localhost, port=4560)
# Add a json-formatter
/subsystem=logging/json-formatter=json:add
# Add the socket handler
/subsystem=logging/socket-handler=log-server-handler:add(named-
formatter=json,level=INFO, outbound-socket-binding-ref=log-server)
# Add the handler to the root logger
/subsystem=logging/root-logger=ROOT:add-handler(name=log-server-
handler)
```

*Добавьте пример UDP*

```
/subsystem=logging/socket-handler=log-server-handler:add(named-
formatter=json,level=INFO, outbound-socket-binding-ref=log-server,
protocol=UDP)
```

*Добавить пример SSL*

```
# Add the socket binding
/socket-binding-group=standard-sockets/remote-destination-outbound-
socket-binding=logserver:add(host=localhost, port=4560)
# Add the Elytron key store
/subsystem=elytron/key-store=log-server-
ks:add(path=/path/to/keystore.jks, type=JKS,credential-
reference={clear-text=mypassword})
# Add the Elytron trust manager
/subsystem=elytron/trust-manager=log-server-tm:add(key-store=log-
server-ks)
# Add the client SSL context
/subsystem=elytron/client-ssl-context=log-server-context:add(trust-
manager=log-servertm,protocols=["TLSv1.2"])
# Add a json-formatter
/subsystem=logging/json-formatter=json:add
# Add the socket handler
/subsystem=logging/socket-handler=log-server-handler:add(named-
formatter=json,level=INFO, outbound-socket-binding-ref=log-server,
protocol=SSL_TCP, ssl-context=logserver-context)
# Add the handler to the root logger
/subsystem=logging/root-logger=ROOT:add-handler(name=log-server-
handler)
```

**Внимание:** преобразование обработчика сокета в асинхронный обработчик может повысить производительность;

– обработчик системного журнала (syslog-handler) – это обработчик, который выполняет запись на сервер системного журнала через UDP. Обработчик поддерживает форматы RFC3164 или RFC5424.

**Внимание:** в обработчике системного журнала отсутствуют некоторые свойства конфигурации, которые могут быть полезны в некоторых сценариях, например, при настройке форматирования. Используйте «org.jboss.logmanager.handlers.SysLogHandler» в модуле «org.jboss.logmanager» в качестве пользовательского обработчика для использования этих преимуществ. В какой-то момент будут добавлены дополнительные атрибуты, так что в этом больше не будет необходимости.

### 7.5.9 Как

***– Как добавить категорию журнала?***

```
/subsystem=logging/logger=com.your.category:add
```

– ***Как изменить уровень ведения журнала?***

Чтобы изменить уровень журнала обработчиков, выполните следующие действия:

```
/subsystem=logging/console-handler=CONSOLE:write-
attribute(name=level,value=DEBUG)
```

Изменение уровня в категории журнала происходит аналогично:

```
/subsystem=logging/logger=com.your.category:write-
attribute(name=level,value=ALL)
```

– ***Как записать сообщения приложений пользователя в их собственный файл?***

1) Создайте обработчик файлов. Существует 3 различных типа обработчиков файлов на выбор: обработчик файлов, обработчик файлов с периодической ротацией и обработчик файлов с ротацией размера. В этом примере использован простой обработчик файлов.

```
/subsystem=logging/file-handler=fh:add(level=INFO,
file={"relativeto"=>"jboss.server.log.dir", "path"=>"fh.log"},
append=false, autoflush=true)
```

2) Теперь создайте категорию журнала.

```
/subsystem=logging/logger=org.your.company:add(use-
parenthandlers=false,handlers=["fh"])
```

– ***Как использовать «log4j.properties» или «log4j.xml» вместо использования конфигурации подсистемы ведения журнала?***

Прежде всего, обратите внимание, если решено использовать файл конфигурации «log4j», больше не будет возможности вносить изменения в конфигурацию ведения журнала развертываний во время выполнения.

Если это приемлемо, вы можете использовать ведение журнала для каждого развертывания и просто включить файл конфигурации в свое развертывание.

– ***Как использовать собственную пользовательскую версию «log4j»?***

Если необходимо включить пользовательскую версию «log4j», нужно выполнить следующие два шага.

1) Отключите добавление зависимостей ведения журнала во все пользовательские развертывания с помощью атрибута «add-loggingapi- dependencies» и отключите атрибут «use-deployment-logging-config» или исключите подсистему ведения журнала в виде «jboss-deployment-structure.xml».

2) Затем необходимо включить библиотеку «log4j» в пользовательском развертывание.

Это работает только для ведения журнала в вашем развертывании. В журналах сервера по-прежнему будет использоваться конфигурация подсистемы ведения журнала.

– ***Как использовать собственную пользовательскую реализацию log4j2?***

Если нужно использовать собственную пользовательскую реализацию log4j2, такую как log4j-core, необходимо выполнить следующие два шага.

1) Отключите добавление зависимостей ведения журнала во все пользовательские развертывания с помощью атрибута «add-loggingapi-dependencies» или исключите «org.apache.logging.log4j.api» в структуре «jboss-deploymentstructure.xml».

2) Затем нужно будет включить в пользовательское развертывание «log4j-api» и библиотеку реализации «log4j2».

**Внимание:** это работает только для ведения журнала в вашем развертывании. В журналах сервера по-прежнему будет использоваться конфигурация подсистемы ведения журнала.

#### 7.5.10 Регистраторы

##### 7.5.10.1 WIP

**Внимание:** работа над этим еще продолжается. Пожалуйста, не стесняйтесь исправлять любые обнаруженные ошибки

##### 7.5.10.2 Обзор

Логгеры используются для регистрации сообщений. Логгер определяется категорией, обычно состоящей из имени пакета или класса.

Регистратор - это первый шаг к определению того, следует ли регистрировать сообщения или нет. Если для регистратора задан уровень, то уровень сообщения должен быть больше, чем уровень, определенный в регистраторе. Затем выполняется следующая проверка фильтра, и правила фильтра будут определять, являются ли сообщения доступными для регистрации или нет.

***Ресурс регистратора***

Ресурс регистратора использует путь «subsystem=logging/logger=$category», где «$category» - путь регистратора. Например, для регистратора с именем «org.WildBoss Pro.example» был бы указан путь к ресурсу «subsystem=logging/logger=org.WildBoss Pro.example».

Регистратор в виде 4 (четырёх) доступных для записи атрибутов:

– `filter-spec` - это строка на основе выражения для определения фильтров для регистратора.

**Внимание:** фильтры в регистраторах не наследуются;

– `#handlers` - это список имен обработчиков, которые должны быть прикреплены к регистратору. Если атрибуту «useparent-handlers» присвоено значение «true» и сообщения журнала доступны для регистрации, родительские регистраторы будут продолжать обрабатываться;

– `#level` - задает минимальный уровень, на котором можно регистрировать сообщения для регистратора;

– `use-parent-handlers` - это логический атрибут, который определяет, должны ли родительские регистраторы также обрабатывать сообщение журнала.

**Внимание:** пользователь может заметить, что атрибуты «category» и «filter» отсутствуют. Хотя «filter» доступен для записи, в будущем он может быть признан устаревшим и удален. Оба атрибута по-прежнему присутствуют на ресурсе по устаревшим причинам.

***Ресурс корневого регистратора***

Корневой регистратор похож на ресурс «#Logger», только у него нет категории и его имя должно быть «ROOT».

***Иерархия регистраторов***

Иерархия логгера определяется его категорией. Категория представляет собой строку, разделенную точкой, обычно состоящую из имени пакета или класса. Например, логгер «org.WildBoss Pro» является родительским логгером для «org.WildBoss Pro.example».

### 7.5.11 Фильтры ведения журнала

Фильтры используются для обеспечения детального контроля над сообщением журнала. Фильтр может быть назначен регистратору или обработчику журнала. Подробную информацию о фильтрах смотрите в документации по фильтрам.

#### 7.5.11.1 Фильтр

Ресурс «filter» позволяет использовать пользовательский фильтр. Пользовательский фильтр должен находиться в модуле и реализовывать интерфейс фильтра.

Обычно рекомендуется добавлять фильтры в обработчик. По умолчанию логгеры не наследуют фильтры. Это означает, что если для регистратора с именем «org.jboss.as» установлен фильтр, то регистрация выполняется только в том случае, если имя регистратора равно «org.jboss.as.logging».

***Примеры***

*Добавление фильтра*

```
/subsystem=logging/filter=myFilter:add(class=org.jboss.example.MyFilter
,module=org.jboss.example, properties={matches="true"},
constructorproperties={pattern="*.WFLYLOG.*"))
```

*Вложение фильтра*

```
/subsystem=logging/console-handler=CONSOLE:write-attribute(name=filter-
spec,value=not(myFilter))
```

#### 7.5.11.2 Фильтруйте выражения

| Тип фильтра | Выражение | Описание | Параметр(ы) | Примеры |
|---|---|---|---|---|
| accept | accept | Принимает все сообщения журнала | Никто (нет) | accept |
| deny | deny | Отклоняет все сообщения журнала | Никто (нет) | deny |
| not | not(filter Expression) | Принимает фильтр в качестве аргумента и инвертирует возвращаемое значение | Выражение принимает в качестве аргумента один фильтр | not(match("JBAS")) |
| all | all(filter Expressions) | Фильтр, состоящий из нескольких фильтров в цепочке. Если какой-либо фильтр обнаружит, что сообщение журнала невозможно зарегистрировать, сообщение не будет зарегистрировано, а последующие фильтры проверяться не будут | В качестве аргумента выражение принимает список фильтров, разделенных запятыми | all(match("JBAS"), match("WELD")) |
| any | any(filter Expressions) | Фильтр, состоящий из нескольких фильтров | В качестве аргумента | any(match("JBAS"), match |

| Тип фильтра | Выражение | Описание | Параметр(ы) | Примеры |
|---|---|---|---|---|
| | | в цепочке. Если какой-либо фильтр сделает сообщение журнала доступным для регистрации, сообщение будет зарегистрировано, а последующие фильтры проверяться не будут | выражение принимает список фильтров, разделенных запятыми | ("WELD")) |
| levelChange | levelChange (level) | Фильтр, который изменяет запись журнала на новый уровень | В качестве аргумента выражение принимает один строковый уровень | levelChange (WARN) |
| levels | levels(levels) | Фильтр, который включает сообщения журнала с уровнем, указанным в списке уровней | В качестве аргумента выражение принимает список уровней на основе строк, разделенных запятыми | levels(DEBUG, INFO, WARN, ERROR) |
| levelRange | levelRange([ min Level,maxLevel]) | Фильтр, который регистрирует записи, находящиеся в пределах заданного диапазона уровней | В выражении фильтра используется "[" для указания минимального уровня включения и "]" для указания максимального уровня включения. В противном случае используйте "(" или ")", соответственно, для обозначения исключающего. Первым аргументом для выражения является минимальное значение допустимый уровень, второй аргумент - максимальный допустимый уровень | минимальный уровень должен быть меньше, чем ERROR, а максимальный уровень должен быть больше, чем DEBUGlevelRange[ ERROR, DEBUG) минимальный уровень должен быть меньше или равен ERROR, а максимальный уровень должен быть больше, чем DEBUGlevelRange[ ERROR, DEBUG) минимальный |

| Тип фильтра | Выражение | Описание | Параметр(ы) | Примеры |
|---|---|---|---|---|
| | | | | уровень должен быть меньше или равен ERROR , а максимальный уровень должен быть больше или равен INFOlevelRange[ERROR, INFO] |
| match | match("pattern") | Фильтр, основанный на обычном выражении. Необработанное неформатированное сообщение используется в соответствии с шаблоном | В качестве аргумента выражение принимает регулярное выражение. match("JBAS\d+") | |
| substitute | substitute("pattern", "replacement value") | Фильтр, который заменяет первое совпадение с шаблоном на заменяющее значение | Первым аргументом для выражения является шаблон, вторым аргументом - заменяющий текст | substitute("JBAS", "EAP") |
| substituteAll | substituteAll("pattern", "replacement value") | Фильтр, который заменяет все совпадения шаблона на заменяющее значение | Первым аргументом для выражения является шаблон, вторым аргументом - заменяющий текст | substituteAll("JBAS", "EAP") |
| *filterName* | myCustomFilter | Пользовательский фильтр, определенный в ресурсе фильтра | Никто (нет) | myCustomFilterany(myFilter1, myFilter2, myFilter3) |

## 7.6 Руководство по настройке подсистемы EJB3

На этой странице перечислены параметры, доступные для настройки подсистемы EJB. Ниже приведен полный пример конфигурации с полным объяснением каждого из них.

```
<subsystem xmlns="urn:jboss:domain:ejb3:8.0">
    <session-bean>
        <stateless>
            <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
        </stateless>
        <stateful default-session-timeout="600000" default-access-timeout="5000"
        cacheref="
        simple" clustered-cache-ref="clustered"/>
        <singleton default-access-timeout="5000"/>
    </session-bean>
    <mdb>
        <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
        <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
    </mdb>
    <entity-bean>
        <bean-instance-pool-ref pool-name="entity-strict-max-pool"/>
    </entity-bean>
    <pools>
        <bean-instance-pools>
            <strict-max-pool name="slsb-strict-max-pool" max-pool-size="20"
            instanceacquisition-
            timeout="5" instance-acquisition-timeout-unit="MINUTES"/>
            <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20"
            instanceacquisition-
            timeout="5" instance-acquisition-timeout-unit="MINUTES"/>
            <strict-max-pool name="entity-strict-max-pool" max-pool-size="100"
            instanceacquisition-
            timeout="5" instance-acquisition-timeout-unit="MINUTES"/>
        </bean-instance-pools>
    </pools>
    <caches>
        <cache name="simple" aliases="NoPassivationCache"/>
        <cache name="passivating" passivation-store-ref="file" aliases=
        "SimpleStatefulCache"/>
        <cache name="clustered" passivation-store-ref="infinispan" aliases=
        "StatefulTreeCache"/>
    </caches>
    <passivation-stores>
        <file-passivation-store name="file"/>
```

```
        <cluster-passivation-store name="infinispan" cache-container="ejb"/>
    </passivation-stores>
    <async thread-pool-name="default"/>
    <timer-service thread-pool-name="default" default-data-store="default-file-store">
        <data-stores>
            <file-data-store name="default-file-store" path="timer-service-data" relativeto="
            jboss.server.data.dir"/>
        </data-stores>
    </timer-service>
    <remote connectors="remoting-connector" thread-pool-name="default">
        <profiles>
            <profile name="profile" exclude-local-receiver="false" local-receiver-pass-
            byvalue="
            false">
            <remoting-ejb-receiver name="receiver" outbound-connection-ref="remote-
            ejbconnection"/>
            <remote-http-connection name="connection" uri=
            "http://localhost:8180/WildBoss Pro-services"/>
            <static-ejb-discovery>
                <module uri="http://localhost:8180/WildBoss Pro-services" module-
                name="foo"
                app-name="bar" distinct-name="baz"/>
            </static-ejb-discovery>
            </profile>
        </profiles>
    </remote>
    <thread-pools>
        <thread-pool name="default">
            <max-threads count="10"/>
            <core-threads count="10"/>
            <keepalive-time time="60" unit="seconds"/>
        </thread-pool>
    </thread-pools>
    <iiop enable-by-default="false" use-qualified-name="false"/>
    <in-vm-remote-interface-invocation pass-by-value="false"/> <!-- Warning see notes
    below about possible issues -->
</subsystem>
```

### 7.6.1 Компонент сеанса (<session-bean>)

Элемент <stateless> используется для настройки пула экземпляров, который по умолчанию используется для компонентов сеанса без сохранения состояния. Если его нет, компоненты сеанса без сохранения состояния не объединяются в пул, а вместо этого создаются по требованию для каждого вызова. Пул экземпляров может быть переопределен на уровне каждого развертывания или компонента с помощью «jboss-ejb3.xml» или аннотация «org.jboss.ejb3.annotation.Pool». Сами пулы экземпляров настраиваются в элементе <pools>.

Элемент <stateful> используется для настройки компонентов сеанса с отслеживанием состояния:

– «default-session-timeout» этот необязательный атрибут определяет количество времени по умолчанию в миллисекундах, в течение которого компонент сеанса с отслеживанием состояния может оставаться бездействующим, прежде чем он будет удален контейнером. Его можно переопределить с помощью «ejb-jar.xml» дескриптора развертывания или аннотации «javax.ejb.StatefulTimeout»;

– «default-session-timeout» этот необязательный атрибут определяет количество времени по умолчанию в миллисекундах, в течение которого компонент сеанса с отслеживанием состояния может оставаться бездействующим, прежде чем он будет удален контейнером. Его можно переопределить с помощью «ejb-jar.xml» дескриптора развертывания или аннотации «javax.ejb.StatefulTimeout»;

– «default-session-timeout» этот необязательный атрибут определяет количество времени по умолчанию в миллисекундах, в течение которого компонент сеанса с отслеживанием состояния может оставаться бездействующим, прежде чем он будет удален контейнером. Его можно переопределить с помощью «ejb-jar.xml» дескриптора развертывания или аннотации «javax.ejb.StatefulTimeout»;

– «clustered-cache-ref» этот атрибут используется для установки кэша по умолчанию для кластеризованных компонентов.

Элемент <singleton> используется для настройки одноэлементных сеансовых компонентов:

– «default-access-timeout» этот атрибут определяет время, в течение которого одновременные вызовы по умолчанию будут ожидать получения блокировки экземпляра. Его можно переопределить с помощью дескриптора развертывания или аннотации «javax.ejb.AccessTimeout».

### 7.6.2 <mdb>

Элемент <resource-adaptor-ref> устанавливает адаптер ресурсов по умолчанию для компонентов, управляемых сообщениями.

Элемент <bean-instance-pool-ref> используется для настройки пула экземпляров, который по умолчанию используется для управления сообщениями Зернышки. Если он отсутствует, они не объединяются в пул, а вместо этого создаются по требованию для каждого вызова. Пул экземпляров может быть переопределен на уровне каждого развертывания или компонента с помощью «jbossejb3.xml» или аннотация «org.jboss.ejb3.annotation.Pool». Сами пулы экземпляров настраиваются в элементе <pools>.

### 7.6.3 Объект - компонент (<entity-bean>)

Элемент <entity-bean> используется для настройки поведения для EJB2 EntityBeans.

Элемент <bean-instance-pool-ref> используется для настройки пула экземпляров, который используется по умолчанию для компонентов Entity Beans. Если его нет, они не объединяются в пул, а вместо этого создаются по требованию для каждого вызова. Пул экземпляров может быть переопределен на уровне каждого развертывания или

компонента с помощью «jboss-ejb3.xml» или аннотация «org.jboss.ejb3.annotation.Pool». Сами пулы экземпляров настраиваются в элементе <pools>.

### 7.6.4 <pools>

### 7.6.5 <caches>

### 7.6.6 <passivation-stores>

### 7.6.7 <async>

Элемент <async> позволяет выполнять асинхронные вызовы EJB. Он также используется для указания пула потоков, который будут использовать эти вызовы.

### 7.6.8 <timer-service>

Элемент <timer-service> включает службу EJB timer. Он также используется для указания пула потоков, который будут использовать эти вызовы.

Элемент <data-store> используется для настройки каталога, в который сохраняется постоянная информация о таймере.

### 7.6.9 <remote>

Элемент <remote> используется для включения удаленных вызовов EJB. Здесь указывается разделенный пробелами список подключений удаленного взаимодействия, которые будут использоваться (как определено в конфигурации подсистемы удаленного взаимодействия), и пул потоков, который будет использоваться для удаленных вызовов.

Удаленный профиль <profile> определяет конфигурацию удаленных вызовов, на которые могут ссылаться многие развертывания. Файлы EJB, которые предназначены для вызова, могут быть обнаружены как статическим, так и динамическим способом.

Статическое обнаружение определяет, к какому удаленному узлу подключиться, на основе информации, предоставленной администратором.

«Dynamic discovery» отвечает за мониторинг доступных EJBS на всех узлах, к которым настроены подключения, и принимает решение о том, к какому удаленному узлу подключиться, на основе собранных данных:

– назовите название профиля;

– если параметр «exclude-local-receiver» установлен, то в этом профиле локальный приемник не используется;

– если задано «local-receiver-pass-by-value», локальный приемник будет передавать ejb-компоненты по значению.

Статическое обнаружение «ejb» позволяет администратору явно указать, на каких удаленных узлах расположены данные EJB . Для его определения используется тег «module»:

– «module-name» - название модуля EJB;

– «app-name» - название приложения EJB;

– «distinct» -присвоить различное имя EJB;

– «uri» - адрес, по которому находится данный EJB.

Тег «remoting-ejb-receiver» используется для определения динамического обнаружения на основе протокола удаленного взаимодействия:

– «name» - имя удаленного подключения;

– «app-name» - название приложения EJB;

– «distinct» -присвоить различное имя EJB;

– «uri» - адрес, по которому находится данный EJB.

Тег «remoting-ejb-receiver» используется для определения динамического обнаружения на основе протокола удаленного взаимодействия:

– «name» - имя удаленного подключения;

– «outbound-connection-ref» - ссылка на исходящее соединение, определенное в подсистеме удаленного взаимодействия;

– connection-timeout - время ожидания соединения.

Тег удаленного http-соединения используется для определения динамического обнаружения на основе протокола HTTP:

– «name» - имя HTTP-соединения;

– «uri» - URI соединения.

### 7.6.10 <thread-pools>

Элемент <thread-pools> используется для настройки пулов потоков, используемых при асинхронных, таймерных и удаленных вызовах:

– «max-threads» указывает максимальное количество потоков в пуле потоков. Это обязательный атрибут, значение по умолчанию равно 10;

– параметр «core-threads» определяет количество основных потоков в пуле потоков. Это необязательный атрибут, по умолчанию используется значение «max-threads».;

– значение «keepalive-time» определяет период времени, в течение которого неосновные потоки могут оставаться незанятыми, прежде чем их можно будет удалить. Это необязательный атрибут, значение которого по умолчанию равно 60 секундам.

### 7.6.11 <iiop>

Элемент <iiop> используется для включения IIOP (т.е. CORBA) вызова EJB. Если этот элемент присутствует, то также должна быть установлена подсистема JacORB. Она поддерживает следующие два атрибута:

– «enable-by-default» (включить по умолчанию), если это верно, то все EJB с домашними интерфейсами EJB2.x доступны через IIOP, в противном случае они должны быть явно включены через «jboss-ejb3.xml»;

– «use-qualified-name», если это значение равно «true», то EJB привязываются к контексту именования «corba» с помощью имени привязки, которое содержит название приложения и модулей для развертывания (например, myear/myejbjar/MyBean), если это значение равно «false», то имя привязки по умолчанию - это просто имя компонента.

### 7.6.12 <in-vm-remote-interface-invocation>

По умолчанию при вызове удаленного интерфейса используется передача по значению, как того требует спецификация EJB. Этот элемент можно использовать для включения передачи по ссылке, что может повысить производительность. Обратите внимание, что WildBoss Pro выполнит поверхностную проверку, чтобы убедиться, что вызывающий объект и EJB имеют доступ к одним и тем же определениям классов, что означает, что, если вы передаете что-то вроде списка <MyObject>, WildBoss Pro проверяет список только для того, чтобы убедиться, что это одно и то же определение класса на стороне вызова и EJB. Если определение класса верхнего уровня совпадает, JBoss выполнит вызов, используя передачу по ссылке, что означает, что, если MyObject или любые объекты под ним загружаются из разных загрузчиков классов, вы получите исключение ClassCastException. Если определения классов верхнего уровня загружаются из разных загрузчиков классов, JBoss будет использовать передачу по значению. JBoss не может выполнить глубокую проверку всех классов, чтобы гарантировать отсутствие исключений ClassCastExceptions, поскольку выполнение глубокой проверки исключило бы любое повышение производительности, которое вы получили бы, используя вызов по ссылке.

Чтобы настроить передачу по ссылке для вызывающего абонента/клиента, вы настраиваете развертывание верхнего уровня с помощью «META-INF/jboss-ejb-client.xml» содержащего:

```
<jboss-ejb-client xmlns="urn:jboss:ejb-client:1.0">
  <client-context>
    <ejb-receivers local-receiver-pass-by-value="false"/>
  </client-context>
</jboss-ejb-client>
```

### 7.6.13 <server-interceptors>

Элемент <server-interceptors> настраивает ряд серверных перехватчиков, которые можно настроить без изменения развертываний.

Каждый перехватчик настраивается в теге <interceptor>, который содержит следующие поля:

– «module» - модуль, в котором определен перехватчик;

– «class» - класс, который реализует перехватчик.

Чтобы использовать серверные перехватчики, вам необходимо создать модуль, который их реализует, и поместить его в каталог «${WILDBOSS PRO_HOME}/modules».

Реализации перехватчика - это классы POJO, которые используют «javax.interceptor.AroundInvoke» и «javax.interceptor.AroundTimeout» для обозначения методов перехватчика.

Примерная конфигурация:

```
<server-interceptors>
    <interceptor module="org.foo:FooInterceptor:1.0"
    class="org.foo.FooInterceptor"/>
</server-interceptors>
```

Пример реализации перехватчика:

```
package org.foo;
import javax.annotation.PostConstruct;
import javax.interceptor.AroundInvoke;
import javax.interceptor.InvocationContext;
public class FooInterceptor {
  @AroundInvoke
  public Object bar(final InvocationContext invocationContext) throws
  Exception {
    return invocationContext.proceed();
  }
 }
```

### 7.6.14 <client-interceptors>

Элемент <client-interceptors> настраивает ряд перехватчиков на стороне клиента, которые можно настроить без изменения развертываний.

Каждый перехватчик настраивается в теге <interceptor>, который содержит следующие поля:

– «module» - модуль, в котором определен перехватчик;

– «class» - класс, который реализует перехватчик.

Чтобы использовать серверные перехватчики, необходимо создать модуль, который их реализует, и поместить его в каталог «${WILDBOSS PRO_HOME}/modules».

В реализациях перехватчика должен быть реализован интерфейс «*org.jboss.ejb.client.EJBClientInterceptor*».

Пример конфигурации:

```
<client-interceptors>
  <interceptor module="org.foo:FooInterceptor:1.0"
  class="org.foo.FooInterceptor"/>
</client-interceptors>
```

Пример реализации перехватчика:

```
package org.foo;
import org.jboss.ejb.client.EJBClientInterceptor;
import org.jboss.ejb.client.EJBClientInvocationContext;
public class FooInterceptor implements EJBClientInterceptor {
   @Override
   public void handleInvocation(EJBClientInvocationContext context)
   throws Exception
     {
        context.sendRequest();
     }
   @Override
   public Object handleInvocationResult(EJBClientInvocationContext
   context) throws
Exception {
         return context.getResult();
         }
}
```

**Внимание:** ссылки в этом документе на Enterprise JavaBeans(EJB) относятся к Jakarta Корпоративные компоненты, если не указано иное.

### 7.7 Конфигурация подсистемы подводного течения

```
Веб-подсистема была заменена в WildBoss Pro 8 на Undertow.
```

Подсистема «undertow» состоит из двух основных частей: конфигурации сервера и контейнера сервлетов, а также некоторых вспомогательных элементов. Дополнительные разделы, такие как балансировка нагрузки и отработка отказа, описаны в «Руководстве по обеспечению высокой доступности». Конфигурация по умолчанию «does» подходит для большинства случаев использования и обеспечивает приемлемые параметры производительности.

Требуемое расширение:

```
<extension module="org.WildBoss Pro.extension.undertow" />
```

Пример базовой конфигурации подсистемы:

```
<subsystem xmlns="urn:jboss:domain:undertow:12.0">
  <buffer-cache name="default" buffer-size="1024" buffers-per-
  region="1024" maxregions="10"/>
  <server name="default-server">
    <http-listener name="default" socket-binding="http" />
    <host name="default-host" alias="localhost">
      <location name="/" handler="welcome-content" />
    </host>
  </server>
  <servlet-container name="default" default-buffer-cache="default"
  stack-traceon-error="local-only" >
    <jsp-config/>
    <persistent-sessions/>
  </servlet-container>
  <handlers>
    <file name="welcome-content" path="${jboss.home.dir}/welcome-
    content"directory-listing="true"/>
  </handlers>
</subsystem>
```

| Атрибут | Описание |
|---|---|
| default-server | сервер по умолчанию, используемый для развертывания |
| default-virtual-host | виртуальный хост по умолчанию, используемый для развертывания |
| default-servlet-container | контейнер сервлетов по умолчанию, используемый для развертывания |
| instance-id | идентификатор подводного течения. По умолчанию используется значение «${jboss.node.name}», если не определено |
| obfuscate-session-route | установите значение «true», чтобы указать, что идентификатор экземпляра должен быть скрыт при маршрутизации. Это предотвращает отправку идентификатора экземпляра по HTTP - соединениям при обработке удаленных запросов с помощью HTTP-вызывающего устройства |
| default-security-domain | домен безопасности по умолчанию, используемый веб - развертываниями |
| statistics-enabled | установите значение «true», чтобы включить сбор статистики для Откат подсистемы |

Внимание: при установке «obfuscate-session-route» в значение «true» имя сервера используется в качестве соли в алгоритме хеширования, который затемняет значение «instance-id». По этой причине настоятельно рекомендуется изменить значение сервера с «default-server» на что-то другое, иначе было бы легко восстановить запутанный маршрут до его исходного значения, используя байты «default-server» в качестве соли.

Зависимости от других подсистем: подсистема ввода-вывода.

### 7.7.1 Конфигурация буферного кэша

Буферный кэш используется для кэширования содержимого, такого как статические файлы. Можно настроить несколько буферных кэшей, что позволяет отдельным серверам использовать кэши разного размера. Буферы распределяются по регионам и имеют фиксированный размер. Если вы кэшируете много небольших файлов, то лучше использовать меньший размер буфера. Общий объем используемого пространства можно рассчитать, умножив размер буфера на количество буферов в каждой области и максимальное количество областей.

```
<buffer-caches>
   <buffer-cache name="default" buffer-size="1024" buffers-per-
   region="1024" maxregions="10"/>
</buffer-caches>
```

| Атрибут | Описание |
|---|---|
| buffer-size | Размер буферов. Буферы меньшего размера позволяют использовать пространство более эффективно |
| buffers-per-region | Количество буферов в каждом регионе |
| max-regions | Максимальное количество областей. Определяет максимальный объем памяти, который может быть использован для кэширования |

### 7.7.2 Конфигурация сервера

Сервер представляет собой экземпляр Undertow. В основном он состоит из набора соединителей и некоторых настроенных обработчиков.

```
<server name="default-server" default-host="default-host" servlet-
container="default">
```

| Атрибут | Описание |
|---|---|
| name | имя этого сервера |
| default-host | виртуальный хост, который будет использоваться при поступлении входящего запроса в качестве заголовка no Host: |
| servlet-container | контейнер сервлета, который будет использоваться этим сервером, если только он явно не переопределен при развертывании |

#### 7.7.2.1 Конфигурация соединителя

Undertow предоставляет соединители HTTP, HTTPS и AJP, которые настраиваются для каждого сервера.

*Общие настройки*

Следующие настройки являются общими для всех разъемов:

| Атрибут | Описание |
|---|---|
| socket-binding | Используемая привязка к сокету. Это определяет адрес и порт, которые прослушивает прослушиватель |
| worker | Ссылка на работника XNIO, как определено в Подсистема ввода-вывода. Используемый рабочий объект управляет пулом потоков ввода-вывода и блокирования |
| buffer-pool | Ссылка на пул буферов, определенный в подсистеме ввода -вывода. Эти буферы используются внутри системы для запросов на чтение и запись. В целом их должно быть не менее 8 кб, если только вы не работаете в среде с ограниченным объемом памяти |
| enabled | Если соединитель включен |
| max-post-size | Максимальный разрешенный размер входящих почтовых запросов |
| buffer-pipelined-data | Если ответы на конвейерные запросы HTTP должны быть буферизованы и отправлены за одну запись. Это может повысить |

| Атрибут | Описание |
|---|---|
| | производительность, если используется конвейерная обработка HTTP и количество ответов невелико |
| max-header-size | Максимальный допустимый размер блока заголовка HTTP. В ответах, содержащих слишком много данных в блоке заголовка, запрос завершается и отправляется неверный ответ на запрос |
| max-parameters | Максимальное количество разрешенных параметров запроса или пути. Это ограничение существует для предотвращения DOS-атак на основе коллизий хэшей |
| max-headers | Максимальное количество разрешенных заголовков. Это ограничение существует для предотвращения DOS-атак на основе коллизий хэшей |
| max-cookies | Максимальное количество разрешенных файлов cookie. Это ограничение существует для предотвращения DOS-атак на основе коллизии хэшей |
| allow-encoded-slash | Установите для этого параметра значение true, если вы хотите, чтобы сервер расшифровывал символы косой черты, закодированные в процентах. Вероятно, это плохая идея, поскольку это может иметь последствия для безопасности из-за того, что разные серверы по-разному интерпретируют косую черту. Включите это, только если у вас есть устаревшее приложение, которое требует этого |
| decode-url | Если URL-адрес должен быть декодирован. Если для этого параметра не задано значение «true», то символы, закодированные в процентах, в URL-адресе будут оставлены как есть |
| url-charset | Кодировка для декодирования URL-адреса, на который |
| always-set-keep-alive | Если заголовок «Connection: keep-alive» должен быть добавлен ко всем ответам, даже если это не требуется по спецификации |
| disallowed-methods | Список запрещенных HTTP-методов, разделенных запятыми. ТРАССИРОВКА HTTP по умолчанию отключена |

*HTTP-соединитель*

```
<http-listener name="default" socket-binding="http" />
```

| Атрибут | Описание |
|---|---|
| certificate-forwarding | Если для этого параметра установлено значение «true», то прослушиватель HTTP будет считывать сертификат клиента из заголовка SSL_CLIENT_CERT. Это позволяет использовать аутентификацию по сертификату клиента, даже если сервер не имеет прямого SSL-соединения с конечным пользователем. Это должно быть включено только для серверов за прокси-сервером, которые были настроены так, чтобы всегда устанавливать эти заголовки |
| redirect-socket | Привязка сокета к перенаправлению запросов, которые также требуют обеспечения безопасности |
| proxy-address-forwarding | Если эта функция включена, то для определения адреса однорангового узла будут использоваться заголовки X-Forwarded-For и X-Forwarded-Proto. Это позволяет приложениям, которые |

| Атрибут | Описание |
|---|---|
| | находятся за прокси-сервером, видеть реальный адрес клиента, а не адрес прокси-сервера |

*Прослушиватель HTTPS*

Прослушиватель Https обеспечивает безопасный доступ к серверу. Наиболее важным параметром конфигурации является «security realm», который определяет безопасный контекст SSL.

```
<https-listener name="default" socket-binding="https" security-
realm="ssl-realm" />
```

| Атрибут | Описание |
|---|---|
| security-realm | Область безопасности, используемая для настройки SSL. О том, как ее настроить, смотрите в разделе Примеры области безопасности: Примеры |
| verify-client | Один из следующих вариантов: NOT_REQUESTED, REQUESTED или REQUIRED. Если используется аутентификация сертификата клиента, это должно быть либо REQUESTED, либо REQUIRED |
| enabled-cipher-suites | Список разрешенных имен шифровальных костюмов |

*Слушатель AJP*

```
<ajp-listener name="default" socket-binding="ajp" />
```

### 7.7.2.2 Конфигурация хоста

Элемент «host» соответствует виртуальному хосту.

| Атрибут | Описание |
|---|---|
| name | Имя виртуального хоста |
| alias | Список дополнительных имен хостов, разделенных пробелами, которые должны быть сопоставлены |
| default-web-module | Имя развертывания, которое должно использоваться для обработки запросов, которые ничему не соответствуют |
| queue-requests-on-start | Если запросы должны быть поставлены в очередь при запуске для этого хоста. Если для этого параметра установлено значение «false», вместо этого будет возвращен код ответа по умолчанию. Примечание: если запрошен некорректный запуск, а атрибут «queue-requests-on-start» не установлен, запросы не будут помещаться в очередь, несмотря на значение «true» для свойства по умолчанию. В случае некорректного запуска требуются запросы без постановки в очередь. Однако, если настроен режим без корректировки, но для «queue-requests-on-start» явно задано значение «true», то запросы будут помещаться в очередь, что фактически отключит режим без корректировки для запросов к этому хосту |

*Ведение журнала доступа к консоли*

Каждый хост обеспечивает ведение журнала доступа к консоли, который записывает структурированные данные в формате JSON. Запись выполняется только в стандартный вывод и представляет собой одну строку структурированных данных в формате JSON.

Атрибут «attributes management model» используется для определения того, какие атрибуты «exchange» следует регистрировать. Это похоже на схему, используемую для традиционного ведения журнала доступа. Основное отличие заключается в том, что, поскольку данные структурированы, важно использовать определенные ключи.

Также существует атрибут «metadata», который позволяет добавлять дополнительные метаданные в выходные данные. Значением атрибута является набор произвольных пар ключ/значение. Значения могут включать в себя выражения модели управления, которые должны быть разрешимы при запуске службы журнала доступа к консоли. Значение разрешается один раз при запуске или перезагрузке сервера.

#### 7.7.2.3 Примеры CLI

*add-console-access-logging.cli*

```
/subsystem=undertow/server=default-server/host=default-
host/setting=console-accesslog:add
```

*complex-add-console-access-logging.cli*

```
/subsystem=undertow/server=default-server/host=default-
host/setting=console-accesslog:add(metadata={"@version"="1",
"qualifiedHostName"=${jboss.qualified.host.name:unknown}},
attributes={bytes-sent={},
date-time={key="@timestamp", date-format="yyyy-MM-dd'T'HH:mm:ssSSS"},
remote-host={},
request-line={}, response-header={key-prefix="responseHeader",
names=["Content-Type"]}, response-code={}, remote-user={}})
```

```
{
  "eventSource":"web-access",
  "hostName":"default-host",
  "@version":"1",
  "qualifiedHostName":"localhost.localdomain",
  "bytesSent":1504,
  "@timestamp":"2019-05-02T11:57:37123",
  "remoteHost":"127.0.0.1",
  "remoteUser":null,
  "requestLine":"GET / HTTP/2.0",
  "responseCode":200,
  "responseHeaderContent-Type":"text/html"
}
```

**Внимание:** приведенный выше JSON-файл отформатирован только для удобства чтения. Выходные данные будут представлены в одной строке.

### 7.7.3 Конфигурация контейнера сервлета

Элемент «servlet-container» соответствует экземпляру контейнера сервлета «Undertow». Большинству серверов потребуется только один контейнер сервлета, однако могут быть случаи, когда имеет смысл определить несколько контейнеров (в частности, если вы хотите, чтобы приложения были изолированы, чтобы они не могли отправлять друг другу запросы с

помощью RequestDispatcher. Вы также можете использовать несколько контейнеров сервлетов для обслуживания разных приложений из одного и того же контекстного пути на разных виртуальных хостах).

| Атрибут | Описание |
|---|---|
| allow-non-standard-wrappers | Спецификация сервлета требует, чтобы приложения обрабатывали запрос/ответ только с помощью классов-оболочек, которые являются продолжением классов ServletRequestWrapper и ServletResponseWrapper. Если для этого параметра установлено значение true, то это ограничение снимается |
| default-buffer-cache | Буферный кэш, который используется для кэширования статических ресурсов в сервлете по умолчанию |
| stack-trace-on-error | Может быть либо «all», либо «none», либо только для локального использования. Если установлено значение «none», трассировка стека Undertow никогда не будет отображаться. Если установлено значение «All», трассировка стека всегда будет отображаться (не рекомендуется для производственного использования). Если установлено значение только для локальных адресов, Undertow будет отображать их только для запросов с локальных адресов, где нет заголовков, указывающих на то, что запрос был проксирован. Обратите внимание, что эта функция означает, что вместо страницы ошибок по умолчанию, указанной в «web.xml» |
| default-encoding | Кодировка по умолчанию, используемая для запросов и ответов |
| use-listener-encoding | Если это верно, то кодировка по умолчанию будет такой же, как и у прослушивателя, получившего запрос |
| preserve-path-on-forward | Если это верно, то возвращаемые значения методов getServletPath(), getRequestURL() и getRequestURI() из HttpServletRequest не изменится после вызова функции RequestDispatcher.forward() и будет указывать на исходный запрошенный ресурс. Если значение равно «false», то после вызова RequestDispatcher.forward() они будут указывать на ресурс, на который выполняется переадресация |

#### 7.7.3.1 Настройка страниц сервера в Jakarta

*Настройка сессионных файлов cookie*
Это позволяет изменять атрибуты сессионного файла cookie.

| Атрибут | Описание |
|---|---|
| name | Название файла cookie |
| domain | Домен файлов cookie |
| comment | Комментарий к файлу cookie |
| http-only | Если файл cookie используется только по протоколу HTTP |
| secure | Если файл cookie помечен как защищенный |
| max-age | Максимальный срок хранения файла cookie |

#### 7.7.3.2 Настройка постоянного сеанса

Постоянные сеансы позволяют сохранять данные сеанса при повторном развертывании и перезапуске. Эта функция включается путем добавления элемента «persistent-sessions» в конфигурацию сервера. В основном это функция, используемая во время разработки.

Если путь не указан, то данные сеанса сохраняются в памяти и будут сохраняться только при повторном развертывании, а не при перезапуске.

| Атрибут | Описание |
|---|---|
| path | Путь к данным постоянных сеансов |
| relative-to | Местоположение, к которому относится путь |

### 7.7.4 Прослушиватели AJP

Прослушиватели AJP являются дочерними ресурсами подсистемы «undertow». Они используются с «mod_jk», «mod_proxy» и «mod_cluster» интерфейса Apache httpd. Каждый прослушиватель ссылается на определенную привязку сокета:

```
[standalone@localhost:9999 /] /subsystem=undertow/server=default-
server:read-childrennames(child-type=ajp-listener)
{
   "outcome" => "success",
   "result" => [
   "ajp-listener",
   ]
}
[standalone@localhost:9999 /] /subsystem=undertow/server=default-
server/ajplistener=*:read-resource(recursive=true)
{
   "outcome" => "success",
   "result" => {
     "enabled" => "true",
     "scheme" => "http",
     "socket-binding" => "ajp",
   }
}
```

Для создания нового ajp-прослушивателя требуется, чтобы сначала объявили привязку нового сокета:

```
[standalone@localhost:9999 /] /socket-binding-group=standard-
sockets/socketbinding=ajp:add(port=8009)
```

Затем вновь созданная неиспользуемая привязка сокета может быть использована для создания новой конфигурации соединителя:

```
[standalone@localhost:9999 /] /subsystem=undertow/server=default-
server/ajplistener=myListener:add(socket-binding=ajp, scheme=http,
enabled=true)
```

### 7.7.5 Использование WildBoss Pro в качестве средства балансировки нагрузки

В WildBoss Pro 10 добавлена поддержка использования подсистемы Undertow в качестве средства балансировки нагрузки. WildBoss Pro поддерживает два разных подхода: вы можете либо определить статический балансировщик нагрузки и указать серверные хосты в своей конфигурации, либо использовать его как интерфейс «mod_cluster» и использовать «mod_cluster» для динамического обновления хостов.

#### 7.7.5.1 Общий обзор

WildBoss Pro использует возможности прокси-сервера Undertow для балансировки нагрузки. Undertow подключается к внутренним серверам с помощью встроенного клиента и запросов к прокси-серверам.

Поддерживаются следующие протоколы:

– http;

– ajp;

– http2;

– h2c (открытый текст HTTP2).

Из этих протоколов h2c должен обеспечивать наилучшую производительность, если внутренние серверы поддерживают его.

Прокси-сервер Undertow использует асинхронный ввод-вывод, в запросе задействованы только потоки ввода -вывода, которые отвечают за подключение. Подключение к серверной части выполняется из того же потока, что устраняет необходимость в каких-либо конструкциях потокобезопасности. Если как внешний, так и внутренний серверы поддерживают отправку сообщений серверу и используется протокол HTTP2, то прокси-сервер также поддерживает отправку ответов клиенту. В случаях, когда прокси-сервер и серверная часть поддерживают отправку запроса сервером, но клиент этого не поддерживает, сервер отправит заголовок X-Disable-Push, чтобы сообщить серверной части, что она не должна пытаться отправить этот запрос.

***Профили сервера для балансировки нагрузки***

В WildBoss Pro 11 добавлены профили балансировщика нагрузки как для автономного, так и для доменного режимов.

*Пример*: запустите автономный подсистему балансировки нагрузки

```
# configure correct path to WildBoss Pro installation
WILDBOSS PRO_HOME=/path/to/WildBoss Pro
# configure correct IP of the node
MY_IP=192.168.1.1
# run the load balancer profile
$WILDBOSS PRO_HOME/bin/standalone.sh -b $MY_IP -bprivate $MY_IP -c
standalone-loadbalancer.xml
```

Настоятельно рекомендуется использовать частную/внутреннюю сеть для связи между балансировщиком нагрузки и узлами. Для этого задайте правильный IP-адрес для частного интерфейса (аргумент-bprivate).

*Пример*: запустите рабочий узел

Запустите сервер с профилем HA (или полным профилем HA), в который включен компонент «mod_cluster». Если в вашей среде работает многоадресная рассылка UDP, рабочие программы должны работать "из коробки" без каких-либо изменений. Если это не так, то настройте IP-адрес подсистемы балансировки нагрузки статически.

```
# configure correct path to WildBoss Pro installation
WILDBOSS PRO_HOME=/path/to/WildBoss Pro
# configure correct IP of the node
MY_IP=192.168.1.2
# Configure static load balancer IP address.
# This is necessary when UDP multicast doesn't work in your
environment.
LOAD_BALANCER_IP=192.168.1.1
$WILDBOSS PRO_HOME/bin/jboss-cli.sh <<EOT
embed-server -c=standalone-ha.xml
/subsystem=modcluster/proxy=default:write-attribute(name=advertise,
value=false)
/socket-binding-group=standard-sockets/remote-destination-outbound-
socketbinding=
proxy1:add(host=$LOAD_BALANCER_IP, port=8090)
/subsystem=modcluster/proxy=default:list-add(name=proxies,
value=proxy1)
EOT
# start the woker node with HA profile
$WILDBOSS PRO_HOME/bin/standalone.sh -c standalone-ha.xml -b $MY_IP -
bprivate $MY_IP
```

Опять же, чтобы обезопасить это, пользователи должны настроить личный/внутренний IP-адрес в переменной MY_IP.

#### 7.7.5.2 Использование WildBoss Pro в качестве средства балансировки статической нагрузки

Чтобы использовать WildBoss Pro в качестве статического балансировщика нагрузки, первым шагом является создание обработчика прокси-сервера в подсистеме Undertow. Для целей этого примера мы собираемся предположить, что наш балансировщик нагрузки будет распределять нагрузку между двумя серверами: «sv1.foo.com» и «sv2.foo.com», и будет использовать протокол AJP.

Первым шагом является добавление обработчика обратного прокси-сервера в подсистему Undertow:

```
/subsystem=undertow/configuration=handler/reverse-proxy=my-
handler:add()
```

Затем нам нужно определить привязку исходящих сокетов для удаленных хостов:

```
/socket-binding-group=standard-sockets/remote-destination-outbound-
socketbinding=remote-host1:add(host=sv1.foo.com, port=8009)
/socket-binding-group=standard-sockets/remote-destination-outbound-
socketbinding=remote-host2:add(host=sv2.foo.com, port=8009)
```

и затем мы добавляем их в качестве хостов в обработчик обратного прокси-сервера:

```
/subsystem=undertow/configuration=handler/reverse-proxy=myhandler/
host=host1:add(outbound-socket-binding=remote-host1, scheme=ajp,
instanceid=myroute, path=/test)
/subsystem=undertow/configuration=handler/reverse-proxy=myhandler/
host=host2:add(outbound-socket-binding=remote-host2, scheme=ajp,
instanceid=myroute, path=/test)
```

Теперь нам нужно добавить обратный прокси-сервер к местоположению. Предполагая, что мы используем путь /app:

```
/subsystem=undertow/server=default-server/host=defaulthost/
location=\/app:add(handler=my-handler)
```

Это все, что нужно сделать. Если вы нажмете в своем браузере на кнопку «http://localhost:8080/app», вы сможете увидеть проксируемый контент. Полную информацию обо всех доступных параметрах конфигурации можно найти в справочнике по подсистеме.

## 7.8 Настройка обмена сообщениями

Настройка сервера обмена сообщениями Jakarta выполняется с помощью подсистемы обмена сообщениями «activemq». В этой главе мы рассмотрим часто используемые параметры конфигурации. Для получения более подробного объяснения, пожалуйста, обратитесь к руководству пользователя Artemis (см. «Справочник компонентов»).

### 7.8.1 Требуемое расширение

Параметры конфигурации, рассмотренные в этом разделе, предполагают, что в вашей конфигурации присутствует расширение «org.WildBoss Pro.extension.messaging-activemq». Это расширение не входит в стандартную конфигурацию «standalone.xml» и «standalone-ha.xml» конфигурации , включенные в дистрибутив WildBoss Pro. Однако он включен в конфигурации «standalone-full.xml» и «standalone-full-ha.xml».

Вы можете добавить расширение в конфигурацию без него, либо добавив элемент <extension module="org.WildBoss Pro.extension.messaging-activemq"/> в «xml», либо используя следующую операцию CLI:

```
[standalone@localhost:9990 /]/extension=org.WildBoss
Pro.extension.messaging-activemq:add
```

### 7.8.2 Соединители

Существует три типа соединителей, которые можно использовать для подключения к серверу обмена сообщениями WildBoss Pro Jakarta:

– «invm-connector» может использоваться локальным клиентом (т.е. клиентом, работающим в той же JVM, что и сервер);

– «netty-connector» может использоваться удаленным клиентом (и использует Netty по протоколу TCP для связи);

– «http-connector» может использоваться удаленным клиентом (и использует веб-сервер Undertow для обновления с HTTP-соединение).

### 7.8.3 Фабрики по подключению к системе обмена сообщениями в Jakarta

Существует три основных типа Jakarta Messaging connection-factory, которые зависят от типа используемых соединителей.

Существует также фабрика объединенных подключений, особенность которой в том, что она, по сути, является фасадом конфигурации как для входящих, так и для исходящих соединителей Artemis Jakarta Connectors Адаптер ресурсов. MDB может быть сконфигурирован для использования фабрики объединенных подключений (например, с помощью @ResourceAdapter). В этом контексте MDB использует входящий соединитель Artemis Jakarta Connectors RA. Другие клиенты могут искать фабрику объединенных

соединений в JNDI (или внедрять ее) и использовать ее для отправки сообщений. В этом контексте такой клиент мог бы использовать исходящий соединитель Artemis Jakarta Connectors RA. Фабрика объединенных подключений также является особенной, поскольку:

– он доступен только локальным клиентам, хотя его можно настроить так, чтобы он указывал на удаленный сервер;

– как следует из названия, он объединен в пул и, следовательно, обеспечивает превосходную производительность для клиентов, которые могут его использовать. Размер пула можно настроить с помощью атрибутов «max-pool-size» и «min-pool-size»;

– он должен использоваться только для отправки (т.е. создания) сообщений при поиске в JNDI или введении;

– его можно настроить на использование определенных учетных данных безопасности с помощью атрибутов пользователя и пароля. Это полезно, если удаленный сервер, на который он указывает, защищен;

– ресурсы, приобретенные с его помощью, будут автоматически зачисляться на любые текущие транзакции в Джакарте. Если вы хотите отправить сообщение из компонента Jakarta Enterprise Beans с помощью CMT, то, скорее всего, это и есть фабрика соединений, которую вы хотите использовать, поэтому операция отправки будет атомарно зафиксирована вместе с остальными транзакционными операциями компонента Jakarta Enterprise Beans.

Чтобы было понятно, входящий соединитель Artemis Jakarta Connectors RA (который предназначен для приема сообщений) используется только MDBS и другими компонентами на базе Jakarta Connectors. Он недоступен для традиционных клиентов.

И «connection-factory», и «pooled-connection-factory» ссылаются на объявление соединителя. Сетевой соединитель связан с привязкой к сокету, которая сообщает клиенту, использующему «connection-factory», куда подключаться:

– «connection-factory», ссылающаяся на netty-connector, подходит для использования удаленным клиентом для отправки сообщений на сервер или получения сообщений с сервера (при условии, что «connection-factory» имеет соответствующую экспортированную запись);

– «pooled-connection-factory», найденная в JNDI или внедренная, ссылающаяся на nettyconnector, подходит для использования локальным клиентом для отправки сообщений на удаленный сервер при условии, что привязка сокета ссылается на привязку исходящего сокета, указывающую на соответствующий удаленный сервер;

– «pooled-connection-factory», используемая MDB, которая ссылается на удаленный соединитель, подходит для использования сообщений с удаленного сервера, если привязка сокета ссылается на привязку «outboundsocket», указывающую на соответствующий удаленный сервер.

Встроенный vm-connector связан с идентификатором сервера, который сообщает клиенту, использующему connection-factory , куда подключаться (поскольку несколько серверов Artemis могут работать в одной JVM):

– «connection-factory», ссылающаяся на встроенный в виртуальную машину коннектор, подходит для использования локальным клиентом либо для отправки сообщений на локальный сервер, либо для получения сообщений с него;

– «pooled-connection-factory», найденная в JNDI или внедренная, ссылающаяся на встроенный «vmconnector», подходит для использования локальным клиентом только для отправки сообщений на локальный сервер;

– «pooled-connection-factory», используемая MDB, которая ссылается на «in-vm-connector», подходит только для использования сообщений с локального сервера.

HTTP-коннектор связан с привязкой к сокету, которая представляет HTTP-сокет (по умолчанию именуемый http):

– «connection-factory», ссылающаяся на http-коннектор, подходит для использования удаленным клиентом для отправки сообщений на сервер или получения сообщений с сервера путем подключения к его HTTP-порту перед обновлением до протокола обмена сообщениями;

– «pooled-connection-factory», ссылающаяся на http-коннектор, подходит для использования локальным клиентом для отправки сообщений на удаленный сервер при условии, что привязка сокета ссылается на исходящую привязку сокета, указывающую на соответствующий удаленный сервер;

– «pooled-connection-factory», используемая MDB, которая ссылается на http-коннектор, подходит только для использования сообщений с удаленного сервера , если привязка сокета ссылается на привязку исходящего сокета, указывающую на соответствующий удаленный сервер.

В объявлении объекта «connection-factory» или «pooled-connection-factory» указывается имя JNDI, под которым будет доступна фабрика. Удаленным клиентам доступны только имена JNDI, привязанные к пространству имен «java:jboss/exported». Если у фабрики подключений есть запись, привязанная к пространству имен «java:jboss/exported», удаленный клиент будет искать «connectionfactory», используя текст после «java:jboss/exported». Например, «RemoteConnectionFactory» по умолчанию привязан к «java:jboss/exported/jms/RemoteConnectionFactory», что означает, что удаленный клиент будет искать эту фабрику подключений, используя «jms/RemoteConnectionFactory». У «pooledconnection- factory» не должно быть привязки к какой-либо записи в пространстве имен «java:jboss/exported», поскольку «pooled-connection-factory» не подходит для удаленных клиентов.

Начиная с версии Jakarta Messaging 2.0, для Jakarta доступна фабрика подключений Jakarta Messaging по умолчанию Приложения EE под именем JNDI «java:comp/DefaultJMSConnectionFactory». Подсистема обмена сообщениями WildBoss Pro определяет фабрику объединенных подключений, которая используется для обеспечения этой фабрики подключений по умолчанию. Любое изменение параметров на этой фабрике объединенных подключений будет учтено любым приложением EE, использующим поставщика сообщений Jakarta по умолчанию под именем JNDI java:comp/DefaultJMSConnectionFactory.

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
 <server name="default">
   [...]
   <http-connector name="http-connector"
                   socket-binding="http"
                   endpoint="http-acceptor" />
   <http-connector name="http-connector-throughput"
                   socket-binding="http"
                   endpoint="http-acceptor-throughput">
      <param name="batch-delay"
             value="50"/>
   </http-connector>
   <in-vm-connector name="in-vm"
                    server-id="0"/>
   [...]
   <connection-factory name="InVmConnectionFactory"
                       connectors="in-vm"
                       entries="java:/ConnectionFactory" />
   <pooled-connection-factory name="activemq-ra"
                              transaction="xa"
                              connectors="in-vm"
                              entries="java:/JmsXA
java:jboss/DefaultJMSConnectionFactory"/>
   [...]
 </server>
</subsystem>
```

Видишь standalone/configuration/standalone-full.xml

### 7.8.4 Очереди и темы обмена сообщениями в Jakarta

Очереди и темы обмена сообщениями в Jakarta являются подресурсами подсистемы активного обмена сообщениями. Можно определить либо jms-очередь, либо jms-тему. Каждому получателю должно быть присвоено имя, и в его элементе «entries» должна содержаться хотя бы одна запись (разделенная пробелом).

Каждая запись относится к JNDI-имени очереди или раздела. Имейте в виду, что любая «jms-очередь или jmstopic», доступ к которым требуется удаленному клиенту, должна иметь запись в пространстве имен «java:jboss/exported». Как и в случае с фабриками подключений, если в jms-очереди или jms-теме есть запись, привязанная к пространству имен «java:jboss/exported», удаленный клиент будет искать ее, используя текст после «java:jboss/exported». Например, следующая jms-очередь «testQueue» привязана к «java:jboss/exported/jms/queue/test», что означает, что удаленный клиент будет искать эту «\{{kms-очередь}», используя «jms/queue/test». Локальный клиент может найти его, используя «java:jboss/exported/jms/queue/test», «java:jms/queue/test» или, проще говоря, «jms/queue/test»:

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
   <server name="default">
   [...]
   <jms-queue name="testQueue"
              entries="jms/queue/test
              java:jboss/exported/jms/queue/test" />
   <jms-topic name="testTopic"
             entries="jms/topic/test java:jboss/exported/jms/topic/test"
             />
</subsystem>
```

Видишь standalone/configuration/standalone-full.xml

Конечные точки обмена сообщениями в Jakarta можно легко создать с помощью интерфейса командной строки:

```
[standalone@localhost:9990 /] jms-queue add --queue-address=myQueue
--entries=queues/myQueue
```

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/jmsqueue=myQueue:read-resource
   {
   "outcome" => "success",
   "result" => {
      "durable" => true,
      "entries" => ["queues/myQueue"],
      "selector" => undefined
   }
}
```

Приостановка и возобновление работы очередей и разделов.

Когда очередь приостановлена, она будет получать сообщения, но не будет их доставлять. Когда она возобновится, она начнет доставлять сообщения, находящиеся в очереди, если таковые имеются. Когда тема приостановлена, она будет получать сообщения, но не будет их доставлять. Новые подписчики также будут приостановлены до тех пор, пока

тема не будет возобновлена. Когда она будет возобновлена, рассылка будет осуществляться снова. Параметр «persist» гарантирует, что тема останется приостановленной при перезапуске сервера.

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/jmsqueue=myQueue:pause()
```

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/jmstopic=myTopic:pause()
```

Также доступен ряд дополнительных команд для обслуживания подсистемы обмена сообщениями Jakarta:

```
[standalone@localhost:9990 /] jms-queue --help --commands
add
...
remove
To read the description of a specific command execute 'jms-queue
command_name --help'.
```

### 7.8.5 Отложенное письмо и повторная доставка

Некоторые настройки применяются к подстановочному адресу, а не к конкретному получателю сообщений. К этой группе относятся настройки очереди непринятых писем и повторной доставки:

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
 <server name="default">
   [...]
   <address-setting name="#"
                    dead-letter-address="jms.queue.DLQ"
                    expiry-address="jms.queue.ExpiryQueue"
                    [...] />
```

Видишь standalone/configuration/standalone-full.xml

### 7.8.6 Настройки безопасности для адресов Artemis и пунктов назначения сообщений в Jakarta

Ограничения безопасности сопоставляются с подстановочным знаком адреса, аналогично настройкам DLQ и повторной доставки.

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
  <server name="default">
    [...]
    <security-setting name="#">
      <role name="guest"
            send="true"
            consume="true"
            create-non-durable-queue="true"
            delete-non-durable-queue="true"/>
```

Видишь standalone/configuration/standalone-full.xml

### 7.8.7 Домен безопасности для пользователей

По умолчанию Artemis будет использовать домен безопасности Elytron «ApplicationDomain». Этот домен используется для аутентификации пользователей, подключающихся к Artemis, после чего они получают разрешение на выполнение определенных функций в соответствии с их ролями и настройками безопасности, описанными выше. Этот домен можно изменить, используя домен безопасности, например:

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
 <server name="default">
   <security elytron-domain="mySecurityDomain" />
   [...]
```

### 7.8.8 Кластерная аутентификация

Если сервер Artemis настроен на кластеризацию, он будет использовать атрибуты пользователя и пароля кластера для подключения к другим узлам Artemis в кластере.

Если вы не измените значение по умолчанию <cluster-password>, Artemis не сможет пройти аутентификацию с ошибкой:

```
HQ224018: Failed to create session:
HornetQExceptionerrorType=CLUSTER_SECURITY_EXCEPTION message=HQ119099:
Unable to
authenticate cluster user: HORNETQ.CLUSTER.ADMIN.USER
```

Чтобы предотвратить эту ошибку, необходимо указать значение для <cluster-password>. Это значение можно зашифровать с помощью зашифрованного выражения, обратившись к документации Elytron.

В качестве альтернативы вы можете использовать системное свойство «jboss.messaging.cluster.password», чтобы указать пароль кластера из командной строки.

### 7.8.9 Развертывание - jms.xml файлов

Начиная с версии WildBoss Pro 26.1, у вас есть возможность развернуть -jms.xml файл, определяющий пункты назначения сообщений в Джакарте, например:

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging-deployment xmlns="urn:jboss:messaging-activemq-
deployment:1.0">
  <server name="default">
    <jms-destinations>
     <jms-queue name="sample">
       <entry name="jms/queue/sample"/>
       <entry name="java:jboss/exported/jms/queue/sample"/>
     </jms-queue>
    </jms-destinations>
  </server>
</messaging-deployment>
```

**Внимание:** эта функция в первую очередь предназначена для разработки, поскольку пунктами назначения, развернутыми таким образом, нельзя управлять ни с помощью одного из предоставляемых инструментов управления (например, консоли, интерфейса командной строки и т.д.).

### 7.8.10 Мост обмена сообщениями в Jakarta

Функция Jakarta Messaging bridge заключается в получении сообщений из исходного пункта обмена сообщениями в Джакарте и отправке их целевому пункту обмена сообщениями в Джакарте. Как правило, исходные и целевые пункты назначения находятся на разных серверах. Мост также может использоваться для передачи сообщений с других серверов обмена сообщениями, отличных от Artemis, при условии, что они совместимы с JMS 1.1. Мост обмена сообщениями Jakarta предоставляется проектом Artemis.

Подробное описание доступных свойств конфигурации можно найти в документации по проекту.

#### 7.8.10.1 Модули для других посредников обмена сообщениями

Поиск исходных и целевых ресурсов обмена сообщениями в Джакарте (фабрики назначения и соединений) осуществляется с помощью JNDI. Если исходные или целевые ресурсы управляются другим сервером обмена сообщениями, отличным от WildBoss Pro, необходимые клиентские классы должны быть включены в модуль. Имя модуля должно быть объявлено при настройке Jakarta Messaging Bridge.

Для использования Jakarta Messaging bridge с любым поставщиком сообщений потребуется создать модуль, содержащий «jar» этого поставщика.

Давайте предположим, что мы хотим использовать гипотетического поставщика сообщений с именем AcmeMQ. Мы хотим связать сообщения, поступающие из исходного пункта назначения AcmeMQ, с целевым пунктом назначения на локальном сервере. Сервер обмена сообщениями WildBoss Pro. Для поиска ресурсов AcmeMQ из JNDI требуются 2 jar-файла, acmemq- 1.2.3.jar, mylogapi-0.0.1.jar (пожалуйста, обратите внимание, что этих jar-файлов не существует, это только для примера). Мы не должны включать jar-файл Jakarta Messaging, поскольку он будет предоставляться непосредственно модулем WildBoss Pro.

Чтобы использовать эти ресурсы в Jakarta Messaging bridge, мы должны объединить их в модуле WildBoss Pro, в JBOSS_HOME/modules создаем макет:

```
modules/
`-- org
 `-- acmemq
  `-- main
     |-- acmemq-1.2.3.jar
     |-- mylogapi-0.0.1.jar
     `-- module.xml
```

Мы определяем модуль в «module.xml»:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.9" name="org.acmemq">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>
  <resources>
    <!-- insert resources required to connect to the source or target
    -->
    <!-- messaging brokers if it not another WildBoss Pro instance -->
    <resource-root path="acmemq-1.2.3.jar" />
    <resource-root path="mylogapi-0.0.1.jar" />
  </resources>
  <dependencies>
    <!-- add the dependencies required by messaging Bridge code -->
    <module name="javax.api" /> <!-- deprecated -->
```

```
    <module name="javax.jms.api" />
    <module name="javax.transaction.api"/>
    <module name="org.jboss.remote-naming"/>
    <!-- we depend on org.apache.activemq.artemis module since we will
    send
    messages to -->
    <!-- the Artemis server embedded in the local WildBoss Pro
    instance -->
    <module name="org.apache.activemq.artemis" />
  </dependencies>
</module>
```

#### 7.8.10.2 Конфигурация

Jakarta Messaging bridge определен в разделе «jms-bridge» подсистемы обмена сообщениями «activemq» в файлах конфигурации XML.

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
 <jms-bridge name="myBridge" module="org.acmemq">
   <source connection-factory="ConnectionFactory"
           destination="sourceQ"
           user="user1"
           password="pwd1"
           quality-of-service="AT_MOST_ONCE"
           failure-retry-interval="500"
           max-retries="1"
           max-batch-size="500"
           max-batch-time="500"
           add-messageID-in-header="true">
     <source-context>
       <property name="java.naming.factory.initial"
                 value="org.acmemq.jndi.AcmeMQInitialContextFactory"/>
       <property name="java.naming.provider.url"
                 value="tcp://127.0.0.1:9292"/>
     </source-context>
   </source>
   <target connection-factory"/jms/invmTargetCF"
           destination="/jms/targetQ" />
   </target>
 </jms-bridge>
</subsystem>
```

Разделы «source» и «target» содержат название ресурса обмена сообщениями в Jakarta («connectionfactory» и «destination»), которое будет отображаться в JNDI. В нем дополнительно указываются учетные данные пользователя и пароль. Если они заданы, они будут переданы в качестве аргументов при создании соединения Jakarta Messaging connection из найденного ConnectionFactory. Также можно определить свойства контекста JNDI в разделах «source-context» и «target-context». Если эти разделы отсутствуют, поиск ресурсов обмена сообщениями в Джакарте будет осуществляться в локальном экземпляре WildBoss Pro (как в случае с целевым разделом в примере выше).

#### 7.8.10.3 Команды управления

Мостом обмена сообщениями в Джакарте также можно управлять с помощью интерфейса командной строки WildBoss Pro:

```
[standalone@localhost:9990 /] /subsystem=messaging/jmsbridge=
myBridge/:add(module="org.acmemq",
     source-destination="sourceQ",
     source-connection-factory="ConnectionFactory",
     source-user="user1",
     source-password="pwd1",
     source-context={"java.naming.factory.initial" =>
"org.acmemq.jndi.AcmeMQInitialContextFactory",
                "java.naming.provider.url" => "tcp://127.0.0.1:9292"},
     target-destination="/jms/targetQ",
     target-connection-factory="/jms/invmTargetCF",
     quality-of-service=AT_MOST_ONCE,
     failure-retry-interval=500,
     max-retries=1,
     max-batch-size=500,
     max-batch-time=500,
     add-messageID-in-header=true)
{"outcome" => "success"}
```

Вы также можете просмотреть полное описание ресурса Jakarta Messaging Bridge из командной строки:

```
[standalone@localhost:9990 /] /subsystem=messaging/jms-bridge=*/:read-
resourcedescription
{
   "outcome" => "success",
   "result" => [{
     "address" => [
        ("subsystem" => "messaging"),
        ("jms-bridge" => "*")
     ],
     "outcome" => "success",
     "result" => {
        "description" => "A Jakarta Messaging bridge instance.",
        "attributes" => {
        ...
     }
   }]
}
```

#### 7.8.10.4 Статистика обмена сообщениями в Jakarta

В настоящее время в Jakarta Messaging bridge доступны две статистические данные: количество обработанных сообщений и количество прерванных/откатанных сообщений. Они доступны с помощью следующей команды:

```
/subsystem=messaging/jms-bridge=myBridge:read-attribute(name=message-
count)
{
    "outcome" => "success",
    "result" => 0L
}
/subsystem=messaging/jms-bridge=myBridge:read-attribute(name=aborted-
message-count)
{
    "outcome" => "success",
```

```
    "result" => 0L
}
```

### 7.8.11 Ссылка на компонент

Подсистема обмена сообщениями «activemq» предоставляется проектом Artemis. Для получения подробного описания доступных свойств конфигурации, пожалуйста, обратитесь к документации проекта.

```
• Artemis Homepage: http://activemq.apache.org/artemis/
• Artemis User Documentation: http://activemq.apache.org/artemis/docs.html
```

#### 7.8.11.1 Контроль использования внутренним брокером памяти и дискового пространства

Вы можете настроить использование дискового пространства журналом, используя атрибут «global-max-disk-usage», таким образом блокируя разбиение по страницам и обработку новых сообщений до тех пор, пока не освободится место на диске. Это делается из командной строки:

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default:writeattribute(name=global-max-disk-usage,
value=70)
{
   "outcome" => "success",
   "response-headers" => {
      "operation-requires-reload" => true,
      "process-state" => "reload-required"
   }
}
```

Вы можете определить, с какой периодичностью проверяется использование диска, используя атрибут «disk-scan-period». Аналогичным образом настройте максимальный объем памяти, необходимый для обработки сообщений, используя атрибут «global-max-memory-size», таким образом блокируя обработку новых сообщений до тех пор, пока не освободится место в памяти. Это делается из командной строки:

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default:writeattribute(name=global-max-memory-size,
value=960000000)
{
   "outcome" => "success",
   "response-headers" => {
      "operation-requires-reload" => true,
      "process-state" => "reload-required"
}
}
```

#### 7.8.11.2 Критический анализ брокера

Когда в брокере что-то идет не так, «critical analyzer» может действовать в качестве средства защиты, отключая брокера или виртуальную машину. Если время отклика превышает заданный тайм-аут, брокер считается нестабильным и может быть предпринято действие,

направленное либо на завершение работы брокера, либо на остановку виртуальной машины. В настоящее время в WildBoss Pro это регистрируется только в журнале, но вы можете изменить это поведение, установив для атрибута «critical-analyzer-policy» значение HALT или SHUTDOWN. Для этого «critical analyzer» измеряет время отклика в:

– доставка в очередь (добавление в очередь);

– хранение журналов;

– операции подкачки по страницам.

Вы можете настроить критический анализатор на брокере с помощью интерфейса командной строки. Чтобы отключить критический анализатор, вы можете выполнить следующую команду интерфейса командной строки:

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default:writeattribute(name=critical-analyzer-enabled,
value=false)
{
   "outcome" => "success",
   "response-headers" => {
      "operation-requires-reload" => true,
      "process-state" => "reload-required"
   }
}
```

Вы можете настроить критический анализатор со следующими атрибутами:

– critical-analyzer-enabled;

– critical-analyzer-timeout;

– critical-analyzer-check-period;

– critical-analyzer-policy.

#### 7.8.11.3 Импорт/ экспорт журнала

WildBoss Pro предоставляет операцию по экспорту журнала в файл, который должен запускаться в режиме администратора. Это делается из командной строки:

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default:exportjournal()
{
   "outcome" => "success",
   "result" => "$JBOSS_HOME/standalone/data/activemq/journal-20210125-
   103331692+0100-dump.xml"
}
```

Теперь вы можете импортировать такой файл дампа в обычном режиме, используя команду:

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default:importjournal(file=$FILE_PATH/journal-
20210125-103331692+0100-dump.xml)
{
   "outcome" => "success"
}
```

Если вам нужно устранить неполадки в журнале, вы можете воспользоваться операцией печати данных. Как и операция экспорта, она должна выполняться в режиме администратора. Кроме того, при этом будет отправлен обратно файл, поэтому для отображения или сохранения результата необходимо выполнить операцию прикрепления. Обратите внимание,

что операция отображения не будет работать должным образом, если вы запрашиваете архивированную версию данных.

```
[standalone@localhost:9990 /] attachment display --
operation=/subsystem=messaging
-activemq/server=default:print-data(secret)
ATTACHMENT a69b87f3-ffeb-4596-be51-d73ebdc48b66:

  _ _ _
   / \ ____| |_ ___ __ __(_) _____
  / _ \| _ \ __|/ _ \ \/ | |/ __/
 / ___ \ | \/ |_/ __/ |\/| | |\___ \
/_/ \_\| \__\____|_| |_|_|/___ /
Apache ActiveMQ Artemis 2.16.0
....
```

### 7.8.12 Подключите фабрику объединенных подключений к удаленному серверу Artemis

Подсистема обмена сообщениями «activemq» позволяет настроить ресурс фабрики объединенных подключений, позволяющий локальному клиенту, развернутому в WildBoss Pro, подключаться к удаленному серверу Artemis.

Настройка такой фабрики объединенных подключений выполняется в 3 этапа:

1) создайте привязку к исходящему сокету, указывающую на удаленный сервер обмена сообщениями:

```
/socket-binding-group=standard-sockets/remote-destination-outbound-
socketbinding=remote-artemis:add(host=<server host>, port=61616)
```

2) создайте удаленный соединитель, ссылающийся на привязку к исходящему сокету, созданную на шаге (1):

```
/subsystem=messaging-activemq/remote-connector=remote-
artemis:add(socketbinding=remote-artemis)
```

3) создайте фабрику объединенных подключений, ссылающуюся на удаленный соединитель, созданный на шаге (2):

```
/subsystem=messaging-activemq/pooled-connection-factory=remoteartemis:
add(connectors=[remote-artemis], entries=[java:/jms/remoteCF])
```

В Artemis 1.x используемые темы и очереди имели префикс («jms.topic.» и «jms.queue.»), который добавлялся перед именем адресата. В Artemis 2.x это уже не так, но по соображениям совместимости WildBoss Pro по-прежнему добавляет эти префиксы и указывает Artemis работать в режиме совместимости. Если вы подключаетесь к удаленной Artemis 2.x, возможно, она не находится в режиме совместимости, и, следовательно, старые префиксы могут больше не использоваться. Если вам нужно использовать пункты назначения без этих префиксов, вы можете настроить свою фабрику подключений так, чтобы они не использовались, установив для атрибута enable-amq1-prefix значение «false».

```
/subsystem=messaging-activemq/pooled-connection-factory=remote-
artemis:writeattribute(name="enable-amq1-prefix", value="false")
```

### 7.8.12.1 Очереди и темы обмена сообщениями в Jakarta на удаленном сервере Artemis

Вы также можете добавить очереди и разделы, определенные на удаленном сервере Artemis, которые будут использоваться так, как если бы они были локальными для сервера. Это означает, что вы можете сделать эти удаленные пункты назначения доступными через JNDI так же, как и локальные пункты назначения. Эти пункты назначения определяются вне серверного элемента:

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
  <external-jms-queue name="testQueue"
         entries="jms/queue/test java:jboss/exported/jms/queue/test"/>
  <external-jms-topic name="testTopic"
         entries="jms/topic/test java:jboss/exported/jms/topic/test" />
</subsystem>
```

Конечные точки обмена сообщениями в Jakarta можно легко создать с помощью интерфейса командной строки:

```
[standalone@localhost:9990 /] /subsystem=messaging-activemq/external-
jms-queue=myQueue:read-resource
{
   "outcome" => "success",
   "result" => {
      "entries" => ["queues/myQueue"]
   }
}
```

У вас нет операций для просмотра атрибутов этих пунктов назначения или управления ими.

### 7.8.12.2 Настройка MDB с использованием фабрики объединенных подключений

Когда pooled-connection-factory настроена для подключения к удаленному Artemis, можно настроить компоненты, управляемые сообщениями (MDB), чтобы они использовали сообщения с этого удаленного сервера.

База данных MDB должна быть помечена аннотацией @ResourceAdapter с использованием имени ресурса «pooledconnection- factory».

```
import org.jboss.ejb3.annotation.ResourceAdapter;
   @ResourceAdapter("remote-artemis")
   @MessageDriven(name = "MyMDB", activationConfig = {
   ...
})
public class MyMDB implements MessageListener {
     public void onMessage(Message message) {
     ...
   }
}
```

Если MDB необходимо отправлять сообщения на удаленный сервер, он должен ввести «pooled-connectionfactory», просмотрев его в JNDI, используя одну из его записей.

```
@Inject
@JMSConnectionFactory("java:/jms/remoteCF")
private JMSContext context;
```

*Конфигурация пункта назначения*

MDB также должен указать, от какого адресата он будет получать сообщения. Стандартный способ заключается в определении свойства конфигурации активации «destinationLookup», которое соответствует Поиск JNDI на локальном сервере. Когда MDB подключается к удаленному серверу Artemis, он теперь создает эти привязки локально. Можно использовать подсистему именования, чтобы настроить внешнюю контекстную федерацию так, чтобы она была локальной Привязки JNDI делегируются внешним привязкам. Однако существует более простое решение для настройки пункта назначения при использовании ресурса Artemis Адаптер. Вместо использования JDNI для поиска целевого ресурса Jakarta Messaging, вы можете просто указать имя целевого ресурса (как настроено на удаленном сервере Artemis), используя свойство конфигурации активации назначения и присвоив свойству конфигурации активации «useJNDI» значение «false», чтобы разрешить Artemis отправлять сообщения на сервер. Адаптер ресурсов автоматически создает пункт назначения обмена сообщениями в Jakarta, не требуя никакого поиска в JNDI.

```
@ResourceAdapter("remote-artemis")
@MessageDriven(name = "MyMDB", activationConfig = {
   @ActivationConfigProperty(propertyName = "useJNDI", propertyValue =
   "
   false"),
   @ActivationConfigProperty(propertyName = "destination",
   propertyValue =
   "myQueue"),
   @ActivationConfigProperty(propertyName = "destinationType",
   propertyValue =
   "javax.jms.Queue"),
   @ActivationConfigProperty(propertyName = "acknowledgeMode",
   propertyValue = "Autoacknowledge")
})
public class MyMDB implements MessageListener {
...
}
```

Эти свойства настраивают MDB на использование сообщений из очереди обмена сообщениями Jakarta с именем «myQueue», размещенной на удаленном сервере Artemis. В большинстве случаев такому MDB не нужно искать другие пункты назначения для обработки используемых сообщений, и он может использовать пункт назначения JMSReplyTo, если он определен в сообщении. Если MDB нужны какие-либо другие адресаты обмена сообщениями в Jakarta, определенные на удаленном сервере, он должен использовать JNDI на стороне клиента, следуя документации Artemis, или настроить контекст внешней конфигурации в подсистеме именования (что позволяет внедрять ресурсы обмена сообщениями в Jakarta, используя аннотацию @Resource).

```
[[configuration of a remote destination using annotations]] ====
Configuration of a remote destination using annotations
```

Аннотация @JMSDestinationDefinition может быть использована для создания пункта назначения на удаленной Artemis Server. Это будет работать так же, как и для локального сервера. В дополнение для этого он должен иметь доступ к очереди управления Artemis. Если

ваша очередь удаленного управления Artemis Server не используется по умолчанию, вы можете передать адрес очереди управления в качестве свойства в @JMSDestinationDefinition. Пожалуйста, обратите внимание, что пункт назначения создается удаленно, но не будет удален после того, как развертывание будет отменено / удалено.

```
@JMSDestinationDefinition(
    // explicitly mention a resourceAdapter corresponding to a pooled-
    connectionfactory resource to the remote server
    resourceAdapter = "activemq-ra",
    name="java:global/env/myQueue2",
    interfaceName="javax.jms.Queue",
    destinationName="myQueue2",
       properties = {
              "management-address=my.management.queue",
              "selector=color = 'red'"
       }
)
```

### 7.8.13 Обратная и прямолинейная совместимость

WildBoss Pro поддерживает как обратную, так и прямую совместимость с устаревшими версиями, которые использовали HornetQ в качестве своих посредников обмена сообщениями (таких как JBoss AS7 или WildBoss Pro 8 и 9). Эти два режима совместимости предоставляются проектом ActiveMQ Artemis, который поддерживает ОСНОВНОЙ протокол HornetQ:

– обратная совместимость: клиенты обмена сообщениями WildBoss Pro (использующие Artemis) могут подключаться к устаревшему серверу приложений (работающему под управлением HornetQ);

– прямая совместимость: устаревшие клиенты обмена сообщениями (использующие HornetQ) могут подключаться к WildBoss Pro 26.1 сервер приложений (под управлением Artemis).

#### 7.8.13.1 Прямая совместимость

Прямая совместимость не требует изменения кода в устаревших клиентах обмена сообщениями. Это обеспечивается подсистемой обмена сообщениями WildBoss Pro-activemq и ее ресурсами.

– «legacy-connection-factory» является подресурсом сервера «messaging-activemq» и может использоваться для хранения в JNDI ConnectionFactory на основе HornetQ

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
  <server name="default">
     ...
     <legacy-connection-factory name="legacyConnectionFactory-
     discovery"
     entries=
     "java:jboss/exported/jms/RemoteConnectionFactory"
     ... />
  </server>
</subsystem>
```

– устаревшие пункты назначения обмена сообщениями на основе HornetQ также можно настроить, предоставив атрибут «legacyentries» ресурсу «jms-queue» и «jms-topic»

```
<jms-queue name="myQueue"
           entries="java:jboss/exported/jms/myQueue-new"
           legacy-entries="java:jboss/exported/jms/myQueue" />
<jms-topic name="testTopic"
           entries="java:jboss/exported/jms/myTopic-new"
           legacy-entries="java:jboss/exported/jms/myTopic" />
```

Устаревшие записи должны использоваться устаревшими клиентами (использующими HornetQ), в то время как обычные записи предназначены для Клиенты обмена сообщениями WildBoss Pro 26.1 в Jakarta (с использованием Artemis).

Затем устаревший клиент выполнит поиск в этих устаревших ресурсах обмена сообщениями для взаимодействия с WildBoss Pro. Чтобы избежать каких-либо изменений кода в устаревших клиентах обмена сообщениями, устаревшие записи JNDI должны соответствовать поиску, ожидаемому устаревшим клиентом.

*Миграция*

Во время миграции устаревшая подсистема обмена сообщениями создаст ресурс фабрики устаревших подключений и добавит устаревшие записи в ресурсы «jms-queue» и «jms-topic», если логическому атрибуту «add-legacyentries» присвоено значение «true» для операции миграции. В этом случае устаревшие записи в перенесенной подсистеме обмена сообщениями «activemq» будут соответствовать записям, указанным в устаревшей подсистеме обмена сообщениями, а обычные записи будут созданы с суффиксом «new». Если во время миграции для параметра «add-legacy-entries» установлено значение «false», в подсистеме обмена сообщениями «activemq» не будут созданы устаревшие ресурсы , и клиенты с устаревшими сообщениями не смогут взаимодействовать с Сервера WildBoss Pro 26.1.

#### 7.8.13.2 Обратная совместимость

Обратная совместимость не требует изменения конфигурации на устаревшем сервере. Клиенты WildBoss Pro 26.1 не ищут ресурсы на устаревшем сервере, а используют JNDI на стороне клиента для создания своих ресурсов обмена сообщениями в Jakarta. Затем клиент Artemis от WildBoss Pro может использовать эти ресурсы для связи с устаревшим сервером по основному протоколу HornetQ.

Artemis поддерживает клиентский JNDI для создания ресурсов обмена сообщениями в Jakarta (ConnectionFactory и Destination).

Например, если клиент обмена сообщениями WildBoss Pro 26.1 хочет взаимодействовать с устаревшим сервером, используя очередь с именем «myQueue», он должен использовать следующие свойства для настройки своего JNDI InitialContext:

```
java.naming.factory.initial=org.apache.activemq.artemis.jndi.ActiveMQIn
itialContextFactory
connectionFactory.jms/ConnectionFactory=tcp://<legacy server
address>:5445? \
protocolManagerFactoryStr=org.apache.activemq.artemis.core.protocol.hor
netq.client.Hor
netQClientProtocolManagerFactory
queue.jms/myQueue=myQueue
```

Затем он может использовать имя «jms/ConnectionFactory» для создания Jakarta Messaging ConnectionFactory и «jms/myQueue» для создания очереди сообщений в Jakarta. Обратите внимание, что свойство «protocolManagerFactoryStr=org.apache.activemq.artemis.core.protocol.hornetq.client». Параметр HornetQClien tProtocolManagerFactory является обязательным при указании URL-адреса

устаревшей фабрики подключений , чтобы клиент Artemis JMS мог взаимодействовать с брокером HornetQ на устаревшем сервере.

### 7.8.14 AIO - NIO для журнала обмена сообщениями

Apache ActiveMQ Artemis (как и HornetQ ранее) поставляется с высокопроизводительным журналом. Поскольку Apache ActiveMQ Artemis сам управляет сохранением данных, а не полагается на базу данных или другой сторонний механизм сохранения данных, он очень оптимизирован для конкретных случаев использования обмена сообщениями. Большая часть журнала написана на Java, однако мы абстрагируемся от взаимодействия с реальной файловой системой, чтобы обеспечить возможность различных подключаемых реализаций.

Apache ActiveMQ Artemis поставляется с двумя реализациями:

– **Java NIO -** первая реализация использует стандартный Java NIO для взаимодействия с файловой системой. Это обеспечивает чрезвычайно высокую производительность и работает на любой платформе, где есть среда выполнения Java 6+;

– **Linux Asynchronous IO** - вторая реализация использует тонкую оболочку машинного кода для взаимодействия с библиотекой асинхронного ввода-вывода Linux (AIO). С помощью AIO Apache ActiveMQ Artemis будет вызван обратно, когда данные будут перенесены на диск, что позволит нам полностью избежать явной синхронизации и просто отправить подтверждение о завершении, когда AIO сообщит нам, что данные были сохранены.

Использование AIO, как правило, обеспечивает даже лучшую производительность, чем использование Java NIO.

Журнал AIO доступен только при запуске ядра Linux версии 2.6 или более поздней и после установки «libaio» (если оно еще не установлено). Если AIO не поддерживается в системе, то Artemis выполнит резервное копирование к НИО. Чтобы узнать, какой тип журнала используется эффективно, вы можете выполнить следующую команду, используя jboss-cli:

```
/subsystem=messaging-activemq/server=default:read-
attribute(name=runtime-journal-type)
```

Пожалуйста, обратите внимание, что AIO представлен ASYNCIO в конфигурации модели WildBoss Pro.

Также, пожалуйста, обратите внимание, что AIO будет работать только со следующими файловыми системами: «ext2», «ext3», «ext4», «jfs», «xfs». С другими файловыми системами, например, с NFS, может показаться, что он работает, но в дальнейшем он будет работать медленнее синхронно. Не размещайте журнал в общем доступе NFS!

Следует добавить, что AIO плохо работает с зашифрованными разделами, поэтому вам придется перейти на NIO для них.

Каковы симптомы болезни альцгеймера?

#### 7.8.14.1 Проблема с AIO на WildBoss Pro 10

Если вы видите следующее исключение в вашем файле журнала WildBoss Pro / консоли

```
[org.apache.activemq.artemis.core.server] (ServerService Thread Pool -
- 64) AMQ222010:Critical IO Error, shutting down the server.
file=AIOSequentialFile:/home/WildBoss Pro/WildBoss Pro-
10.0.0.Final/standalone/data/activemq/journal/activemq-data-2.amq,
message=Cannot open
file:The Argument is invalid: java.io.IOException: Cannot open
file:The Argument is invalid
```

```
at org.apache.activemq.artemis.jlibaio.LibaioContext.open(Native
Method)
```

Это означает, что AIO не работает должным образом в вашей системе.

Чтобы использовать NIO вместо этого, выполните следующую команду, используя «jboss-cli»:

```
/subsystem=messaging-activemq/server=default:write-
attribute(name=journal-type,value=NIO)
```

Вам необходимо перезагрузить ваш сервер, и вы должны увидеть следующую трассировку в консоли вашего сервера:

```
INFO [org.apache.activemq.artemis.core.server] (ServerService Thread
Pool -- 64)
AMQ221013: Using NIO Journal
```

#### 7.8.14.2 Проблема AIO на WildBoss Pro 9

```
[org.hornetq.core.server] (ServerService Thread Pool -- 64) HQ222010:
Critical IO
Error, shutting down the server.file=AIOSequentialFile:/home/WildBoss
Pro/WildBoss Pro-9.0.2.Final/standalone/data/messagingjournal/hornetq-
data-1.hq, message=Can't open
file: HornetQException[errorType=NATIVE_ERROR_CANT_OPEN_CLOSE_FILE
message=Can't open
file]
at org.hornetq.core.libaio.Native.init(Native Method)
```

Это означает, что AIO не работает должным образом в вашей системе.

Чтобы использовать NIO вместо этого, выполните следующую команду, используя «jboss-cli»:

```
/subsystem=messaging/hornetq-server=default:write-
attribute(name=journaltype,value=NIO)
```

Вам нужно перезагрузить ваш сервер, и вы увидите следующую трассировку в консоли вашего сервера:

```
INFO [org.hornetq.core.server] (ServerService Thread Pool -- 64)
HQ221013: Using NIO
Journal
```

### 7.8.15 Хранилище JDBC для журнала обмена сообщениями

Сервер Artemis, интегрированный в WildBoss Pro, может быть настроен на использование хранилища JDBC для своего журнала обмена сообщениями вместо журнала на основе файлов.

Серверный ресурс подсистемы обмена сообщениями «activemq» должен настроить свой атрибут «journal-datasource», чтобы иметь возможность использовать хранилище JDBC.

Если этот атрибут не определен, для сервера Artemis будет использоваться обычный журнал на основе файлов.

Значение этого атрибута должно соответствовать источнику данных, определенному в подсистеме источников данных.

Например, если подсистема источников данных определяет источник данных «ExampleDS» в «/subsystem=datasources/data-source=ExampleDS», сервер Artemis может использовать его для своего хранилища JDBC с помощью операции:

```
/subsystem=messaging-activemq/server=default:write-
attribute(name=journal-datasource,
value=ExampleDS)
```

Artemis JDBC store использует SQL-команды для создания таблиц, используемых для хранения информации. Эти SQL-команды могут отличаться в зависимости от типа базы данных. Команды SQL, используемые хранилищем JDBC, находятся в «modules/system/layers/base/org/apache/activemq/artemis/main/artemis-jdbc-store-${ARTEMIS_VERSION}.jar/journal-sql.properties».

Artemis использует различные таблицы JDBC для хранения информации о привязках, постоянных сообщениях и больших сообщениях (подкачка по страницам пока не поддерживается).

Названия этих таблиц можно настроить с помощью таблиц «journal-bindings-table», «journal-messagestable», «journal-page-store-table» и «journal-large-messages-table». Пожалуйста, обратите внимание, что вам необходимо позаботиться о конфигурации базового пула. Вам нужно как минимум четыре соединения:

– один для переплета;

– один для журнала сообщений;

– один для блокировки аренды (если вы используете HA);

– один для общего состояния диспетчера узлов (если вы используете HA).

Таким образом, вы должны определить минимальный размер пула, равный 4 для пула.

Но один факт, который вам необходимо принять во внимание, заключается в том, что для подкачки и отправки больших сообщений может использоваться неограниченное количество потоков. Размер пула, также известный как максимальный размер пула, должен определяться в соответствии с количеством параллельных потоков, которые выполняют операции потоковой передачи страниц/больших сообщений. Для этого не существует определенного правила, поскольку нет соотношения 1:1 между количеством потоков и количеством подключений. Количество подключений зависит от количества потоков, обрабатывающих операции подкачки и отправки больших сообщений, а также от времени, которое вы готовы потратить на установление соединения (см. блокировка-тайм-аут-ожидание-миллисекунды). Когда появляются новые большие сообщения или операции подкачки, они попадают в выделенный поток и пытаются установить соединение, помещаясь в очередь до тех пор, пока одно из них не будет готово или не истечет время для его получения, что приведет к сбою.

Вам действительно необходимо адаптировать свою конфигурацию в соответствии с вашими потребностями и протестировать ее в вашей среде в соответствии с документацией по подсистеме конфигурации источника данных, а также выполнить тесты и прогоны производительности перед запуском в эксплуатацию.

*Ссылка*

```
• Artemis JDBC Persistence -
https://activemq.apache.org/components/artemis/documentation/latest/persis
tence.html#configuring-jdbc-persistence
• Документация по подсистеме источников данных WildBoss Pro - DataSource
configuration
• Модель конфигурации источника данных WildBoss Pro
link:wildscribe/subsystem/datasources/xa-datasource/index.html)
```

### 7.8.16 Настройка широковещательной передачи/обнаружения

Каждый сервер Artemis может быть настроен на широковещательную передачу и/или обнаружение других серверов Artemis в кластере. Artemis поддерживает два механизма настройки широковещательной передачи/обнаружения.

#### 7.8.16.1 Широковещательная рассылка/обнаружение на основе JGroups

Artemis может использовать членство в существующем канале JGroups как для трансляции своих идентификационных данных, так и для обнаружения узлов, на которых развернуты серверы Artemis. Профиль WildBoss Pro full-ha по умолчанию использует этот механизм для широковещательной рассылки/обнаружения с использованием канала JGroups сервера по умолчанию (как определено подсистемой JGroups).

Чтобы добавить эту поддержку в профиль, который не включает ее по умолчанию, используйте следующее:

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/broadcastgroup=bg-group1:add(jgroups-
cluster=activemq-cluster,connectors=http-connector)
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/discoverygroup=dg-group1:add(jgroups-
cluster=activemq-cluster)
```

Чтобы разделить серверы Artemis, используйте отдельное членство, настройте широковещательную рассылку/обнаружение с использованием отдельного канала. Для этого сначала создайте ресурс канала:

```
[standalone@localhost:9990 /]
/subsystem=jgroups/channel=messaging:add(stack=tcp)
```

Это создаст новый ресурс канала JGroups на основе стека протоколов "tcp". Теперь создайте свои группы вещания/обнаружения, используя этот канал:

```
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/broadcastgroup=bg-group2:add(jgroups-
channel=messaging, jgroups-cluster=activemq-cluster,connectors=http-
connector)
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/discoverygroup=dg-group2:add(jgroups-
channel=messaging, jgroups-cluster=activemq-cluster)
```

#### 7.8.16.2 Многоадресная трансляция/обнаружение

Чтобы передавать идентификационные данные автономным клиентам обмена сообщениями, вы можете дополнительно настроить широковещательную рассылку/обнаружение с помощью многоадресных сокетов.

```
[standalone@localhost:9990 /] /socket-binding-group=standard-
sockets/socketbinding=messaging(interface=private, multicast-
address=230.0.0.4, multicastport=45689)
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/broadcastgroup=bg-group3:add(socket-
binding=messaging, connectors=http-connector)
[standalone@localhost:9990 /] /subsystem=messaging-
activemq/server=default/discovery-group=dg-group3:add(socket-
binding=messaging)
```

#### 7.8.16.3 Кластер за подсистемой балансировки нагрузки HTTP

Если кластер находится за балансировщиком нагрузки HTTP, нам нужно указать клиентам, что они не должны использовать топологию кластера для подключения к нему, а должны продолжать использовать первоначальное подключение к балансировщику нагрузки. Для этого вам нужно указать на фабрике подключений (объединенной в пул) не использовать топологию, установив для атрибута «use-topology-for-load-balancing» значение «false».

```
/subsystem=messaging-activemq/pooled-connection-factory=remote-
artemis:writeattribute(name="use-topology-for-load-balancing",
value="false")
```

#### 7.8.16.4 Сетевая изоляция (разделенный мозг)

Вполне возможно, что если реплицированный оперативный сервер или сервер резервного копирования окажется изолированным в сети, произойдет переход на другой ресурс, и в итоге вы получите 2 оперативных сервера, обслуживающих сообщения в кластере, это мы называем разделенным мозгом. Вы можете устранить эту проблему, настроив один или несколько адресов, являющихся частью вашей сетевой топологии, которые будут проверяться на протяжении всего жизненного цикла сервера.

В этом случае сервер остановится сам, пока сеть не восстановится. Это настраивается с использованием следующих атрибутов конфигурации:

– network-check-NIC - сетевой адаптер (контроллер сетевого интерфейса), который будет использоваться для проверки работоспособности сети;

– network-check-period - частота того, как часто мы должны проверять, работает ли еще сеть;

– network-check-timeout - тайм-аут, использованный при проверке связи;

– network-check-list - это список DNS или IP-адресов, разделенных запятыми, без пробелов (он должен принимать IPV6), который будет использоваться для проверки сети;

– network-check-URL-list - список HTTP URI, которые будут использоваться для проверки сети;

– network-check-ping-command - команда, используемая для проверки связи с IPV4-адресами;

– network-check-ping6-command - команда, используемая для проверки связи с IPV6-адресами.

Например, давайте проверим IP-адрес 10.0.0.1:

```
[standalone@localhost:9990 /]
/subsystem=messaging-activemq/server=default:write-
attribute(name=network-check-list,value="10.0.0.1")
```

Как только версия 10.0.0.1 перестанет отвечать на пинг, вы получите исключение, и брокер остановится:

```
WARN [org.apache.activemq.artemis.logs] (ServerService Thread Pool --
84) AMQ202002:
Ping Address /10.0.0.1 wasnt reacheable.
...
INFO [org.apache.activemq.artemis.logs] (Network-Checker-0
(NetworkChecker))
AMQ201001: Network is unhealthy, stopping service
```

Внимание: убедитесь, что вы понимаете топологию своей сети, поскольку это необходимо для проверки вашей сети. С помощью IP-адреса, которые в конечном итоге могут исчезнуть или стать частично видимыми, могут не сработать. Вы можете использовать список из нескольких IP-адресов. После успешного проверки связи сервер сможет продолжить работу.

## 7.9 Конфигурация подсистемы транзакций

Требуемое расширение:

```
<extension module="org.jboss.as.transactions"/>
```

Пример базовой конфигурации подсистемы:

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
  <core-environment node-identifier="${jboss.tx.node.id:1}">
    <process-id>
      <uuid/>
    </process-id>
  </core-environment>
  <recovery-environment socket-binding="txn-recovery-environment"
                        status-socket-binding="txn-status-manager"/>
  <coordinator-environment statistics-enabled="${WildBoss
  Pro.transactions.statisticsenabled:${WildBoss Pro.statistics-
  enabled:false}}"/>
  <object-store path="tx-object-store"
                relative-to="jboss.server.data.dir"/>
</subsystem>
```

### 7.9.1 Конфигурация транзакционной подсистемы

Подсистема транзакций настраивает поведение менеджера транзакций. Narayana - это менеджер транзакций, используемый в WildBoss Pro. Вторым компонентом, настроенным в

подсистеме, является Клиент транзакций WildBoss Pro (WFTC служит абстрактным уровнем для работы с контекстом транзакций).

#### 7.9.1.1 Настройка компонента Narayana

Структура транзакционной подсистемы соответствует структуре компонента Narayana. Narayana определяет отдельный компонент конфигурации для каждого внутреннего модуля. Например, любая конфигурация , относящаяся к ядру Narayana, доступна через beans CoordinatorEnvironmentBean и CoreEnvironmentBean, для обработки JTA это JTAEnvironmentBean, для настройки восстановления транзакций это RecoveryEnvironmentBean.

Подсистема транзакций предоставляет только часть конфигурации, доступной через Narayana beans. Любые другие параметры конфигурации, предоставляемые Narayana, по-прежнему можно настроить через системные свойства, и обычно требуется перезапуск JVM.

Narayana определяет унифицированные имена для системных свойств, которые используются для настройки. Системное свойство имеет форму [имя компонента].[имя свойства]. Например, системное свойство с именем RecoveryEnvironmentBean.periodicRecoveryInitilizationOffset настраивает время ожидания для первого выполнения периодического восстановления после запуска сервера приложений.

#### 7.9.1.2 Конфигурация в модели и в XML

Подсистема транзакций разделяет конфигурацию на разделы в файле конфигурации XML. Каждый раздел принадлежит некоторому модулю Narayana. Конфигурация для «model», с другой стороны, состоит из плоской структуры атрибутов (большинство из них на верхнем уровне).

Например, подсистема определяет идентификатор узла в элементе «core-environment» XML в конфигурации XML, в то время как атрибут идентификатора узла определяется непосредственно в «/subsystem=transactions resource» в модели.

Описание отдельных атрибутов и их значения можно найти в справочнике по модели Руководство.

**jts** - атрибут модели jts сконфигурирован как XML-элемент jts.

*Конфигурация XML, включающая jts*

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
    ...
    <jts />
    ...
</subsystem>
```

**core-environment** - атрибуты модели «node-identifier», «process-id-uuid», «process-id-socket-binding», «process-id-socket-max-ports» настраиваются в элементе «core-environment» XML.

*Пример конфигурации XML для основной среды*

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
  ...
  <core-environment node-identifier="1">
    <process-id>
       <socket socket-binding="txn-socket-id"
               socket-process-id-max-ports="10"/>
    </process-id>
  </core-environment>
  ...
</subsystem>
```

**recovery-environment** - атрибуты модели «recovery-period», «socket-binding», «recovery-listener», «status-socket-binding» настраиваются в XML-элементе «recovery-environment».

*Пример конфигурации XML для среды восстановления*

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
   ...
   <recovery-environment socket-binding="txn-recovery-environment"
                         status-socket-binding="txn-status-manager"
                         recovery-listener="false" />
   ...
</subsystem>
```

Если вы настроите прослушиватель восстановления, то Narayana свяжет связанный сокет, и пользователь может запросить явный запуск проверки восстановления. В следующем примере мы можем увидеть пример взаимодействия с сокетом.

*связь по telnet со слушателем восстановления*

```
telnet localhost 4712
# command to start the recovery scan
SCAN[enter]
# at this time the transaction recovery has been started
^]
close
```

**coordinator-environment** – «enable-tsm-status», «statistics-enabled», «default-timeout», «maximum-timeout» атрибуты модели настраиваются в соответствии с «coordinator-environment» XML-элемент.

*Пример конфигурации XML для среды coordinator*

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
  ...
  <coordinator-environment enable-tsm-status="true" statistics-
  enabled="true"
                           default-timeout="300" maximum-
  timeout="31536000" />
  ...
</subsystem>
```

**transaction statistics** - когда подсистема определяет значение «true» Narayana, включенное для статистики, она начинает собирать статистику об обработке транзакций. Пользователь может просматривать отдельный атрибут или перечислять все атрибуты статистики в виде группы. Атрибуты статистики транзакций доступны только для чтения во время выполнения.

*Соблюдение всех атрибутов статистики транзакций*

```
# connect to a running application server
./bin/jboss-cli.sh -c
# enable transaction statistics
/subsystem=transactions:write-attribute(name=statistics-enabled,
value=true)
# list all statistics attributes
/subsystem=transactions:read-attribute-group(name=statistics, include-
runtime=true)
```

**object-store** - Narayana необходимо сохранять данные об обработке транзакций в журнале транзакций. Это постоянное хранилище называется «object store» в контексте Narayana. Narayana требуется сохранять журнал для транзакций XA, которые обрабатываются

с использованием протокола двухэтапной фиксации. В противном случае транзакция хранится только в памяти, ничего не сохраняя в хранилище объектов.

Narayana предоставляет три реализации хранилища объектов:

– хранилище ShadowNoFileLock сохраняет записи в структуре каталогов файловой системы. Отдельный файл представляет собой запись журнала подготовленной транзакции. Используется, когда атрибуты «use-jdbc-store» и «use-journal-store» имеют значения «false»;

– хранилище журналов сохраняет записи в файле журнала в файловой системе. Записи хранятся в журнале только для добавления, реализованном в рамках проекта ActiveMQ Artemis. Используется, когда значение атрибута «use-journal-store» равно «true», а значение «use-jdbc-store» равно «false»;

– JDBC хранит постоянные записи в базе данных. Доступ к записям осуществляется через соединение JDBC. Для этого хранилища требуется связанный источник данных из подсистемы источников данных. Используется, когда значение атрибута «usejdbc- store» равно «true», а значение «use-journal-store» равно «false».

**journal object-store** - XML-конфигурация элемента object-store XML, настраивающего хранилище журналов с помощью атрибутов модели object-store-path, object-store-relative-to, journal-store-enable-async-io, представляет собой.

*Пример конфигурации XML для хранилища объектов*

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
   ...
     <object-store path="tx-object-store" relative-
     to="jboss.server.data.dir"/>
     <use-journal-store enable-async-io="true"/>
   ...
</subsystem>
```

**JDBC object-store** - реализация JDBC позволяет сохранять журнал транзакций в базе данных. Подсистема транзакций обращается к базе данных через связанный (через JNDI) нетранзакционный (jta=false) источник данных. Когда подсистема транзакций настраивает реализацию хранилища JDBC, менеджер транзакций создает одну или несколько таблиц базы данных (если они не существуют) для сохранения данных транзакции при запуске WildBoss Pro. Нараяна создает отдельную таблицу для каждого типа хранилища. Нараяна использует тип хранилища для группировки записей транзакций одного типа.

Нараяна использует в WildBoss Pro следующие типы магазинов:

– «action store» хранит данные для транзакций JTA;

– хранилище состояний хранит данные для объектов TXOJ;

– «communications store» хранит данные для мониторинга удаленных транзакций JTS и хранения CORBA Данные IOR.

Конфигурация атрибутов может определять префикс для каждого типа магазина. Когда мы не настраиваем префикс или устанавливаем один и тот же префикс для всех типов магазинов, Narayana сохраняет данные транзакции в той же таблице базы данных. По умолчанию Narayana сохраняет журнал транзакций в таблице базы данных с именем JBossTSTxTable.

*Пример интерфейса jboss cli для настройки хранилища объектов JDBC*

```
# PostgreSQL driver module
./bin/jboss-cli.sh "embed-server, module add --name=org.postgresql
--resources=/tmp/postgresql.jar \
  --dependencies=javax.api\,javax.transaction.api"
# non-jta PostgreSQL datasource creation
./bin/jboss-cli.sh "embed-server --server-config=standalone.xml,data-
source add
--name=JDBCStore \
```

```
  --jndi-name=java:jboss/datasources/jdbcstore_postgresql --jta=false \
  --connection-url=jdbc:postgresql://localhost:5432/test --user-
  name=test
--password=test \
  --driver-name=postgresql"
# transaction subsystem configuration
./bin/jboss-cli.sh "embed-server --server-config=standalone.xml, \
  /subsystem=transactions:write-attribute(name=jdbc-store-datasource, \
  value=java:jboss/datasources/jdbcstore_postgresql), \
  /subsystem=transactions:write-attribute(name=use-jdbc-
  store,value=true)"
./bin/jboss-cli.sh "embed-server --server-config=standalone.xml, \
  /subsystem=transactions:write-attribute(name=jdbc-state-store-
  tableprefix,value=state), \
  /subsystem=transactions:write-attribute(name=jdbc-state-store-
  droptable,value=false),
  /subsystem=transactions:write-attribute(name=jdbc-communication-
  store-tableprefix,value=communication), \
  /subsystem=transactions:write-attribute(name=jdbc-communication-
  store-droptable,value=false),
  /subsystem=transactions:write-attribute(name=jdbc-action-store-
  tableprefix,value=action), \
  /subsystem=transactions:write-attribute(name=jdbc-action-store-
  droptable,value=false)"
```

*Пример конфигурации XML для хранилища объектов JDBC*

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
  ...
    <jdbc-store datasource-jndi-
    name="java:jboss/datasources/jdbcstore_postgresql">
      <action table-prefix="action" drop-table="false"/>
      <communication table-prefix="communication" drop-table="false"/>
      <state table-prefix="state" drop-table="false"/>
    </jdbc-store>
  ...
</subsystem>
```

**commit-markable-resources** - позволяет базе данных, отличной от источника данных XA (т.е. локальному ресурсу), надежно участвовать в Транзакция XA в процессе двухэтапной обработки фиксации. Источник данных должен быть сконфигурирован с подключаемым атрибутом, имеющим значение «true», и связан с подсистемой транзакций в качестве ресурса, помечаемого для фиксации (CMR).

В качестве предварительного условия база данных должна содержать таблицу с именем «xids» (имя таблицы базы данных можно настроить с помощью имени атрибута в разделе «commit-markable-resource»), в которой Narayana сохраняет дополнительные метаданные, когда двухэтапная фиксация подготавливает источник данных, отличный от XA.

SQL select, который должен работать для таблицы «xids», можно найти в коде Narayana.

*Пример инструкции SQL для создания таблицы xids для хранения метаданных CMR*

```
-- PostgreSQL
CREATE TABLE xids (
   xid bytea, transactionManagerID varchar(64), actionuid bytea
);
CREATE UNIQUE INDEX index_xid ON xids (xid);
-- Oracle
CREATE TABLE xids (
   xid RAW(144), transactionManagerID VARCHAR(64), actionuid RAW(28)
);
CREATE UNIQUE INDEX index_xid ON xids (xid);
-- H2
CREATE TABLE xids (
   xid VARBINARY(144), transactionManagerID VARCHAR(64), actionuid
   VARBINARY(28)
);
 CREATE UNIQUE INDEX index_xid ON xids (xid);
```

*Пример настройки источника данных CMR в подсистеме*

```
# parameter 'connectable' is true for datasource
 ./bin/jboss-cli.sh "embed-server --server-config=standalone.xml,\
   /subsystem=datasources/data-
   source=ConnectableCMRDs:add(enabled=true, \
   jndi-name=java:jboss/datasources/ConnectableCMRDs, jta=true, use-
   java-context=true,
\
   use-ccm=true, connectable=true, connection-
   url=\"jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE\", \
   driver-name=h2)"
# linking the datasource into the transaction subsystem
./bin/jboss-cli.sh "embed-server --server-config=standalone.xml,\
/subsystem=transactions/commit-
markableresource=\"java:jboss/datasources/ConnectableCMRDs\":add"
./bin/jboss-cli.sh "embed-server --server-config=standalone.xml, \
   /subsystem=transactions/commit-
   markableresource=\"java:jboss/datasources/ConnectableCMRDs\":write-
   attribute(name=name,value=xids), \
   /subsystem=transactions/commit-
   markableresource=\"java:jboss/datasources/ConnectableCMRDs\":write-
   attribute(name=batch-size,value=10), \
   /subsystem=transactions/commit-
   markableresource=\"java:jboss/datasources/ConnectableCMRDs\":write-
   attribute(name=immediatecleanup,value=false)"
```

*Пример конфигурации XML для ресурсов, помечаемых для фиксации*

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
  ...
    <commit-markable-resources>
      <commit-markable-resource jndi-
      name="java:jboss/datasources/ConnectableCMRDs">
         <xid-location name="xids" batch-size="10"/>
      </commit-markable-resource>
    </commit-markable-resources>
  ...
</subsystem>
```

**log-store** - это ресурс, доступный только во время выполнения, который может быть загружен с моментальным снимком содержимого хранилища объектов Narayana. Операция

«/subsystem=transactions/log-store=log-store:probe» загружает сохраненные записи транзакций из хранилища объектов, которые можно просмотреть в модели. Другое: операция «:probe» очищает старые данные и загружает обновленные записи.

*Изучите моментальный снимок хранилища объектов Narayana*

```
/subsystem=transactions/log-store=log-store:probe
/subsystem=transactions/log-store=log-store:read-
resource(recursive=true, includeruntime=true)
```

Итоговый список будет похож на следующий. В данном случае мы видим одну транзакцию с одним участником со статусом «ПОДГОТОВЛЕНО».

```
{
 "outcome" => "success",
 "result" => {
   "expose-all-logs" => false,
   "type" => "default",
   "transactions" => {"0:ffffc0a80065:-22769d16:60c87436:1a" => {
     "age-in-seconds" => "48",
     "id" => "0:ffffc0a80065:-22769d16:60c87436:1a",
     "jmx-name" => undefined,
     "type" =>
     "StateManager/BasicAction/TwoPhaseCoordinator/AtomicAction",
     "participants" => {"1" => {
       "eis-product-name" => undefined,
       "eis-product-version" => undefined,
       "jmx-name" => undefined,
       "jndi-name" => "1",
       "status" => "PREPARED",
       "type" => "/StateManager/AbstractRecord/XAResourceRecord"
     }}
   }}
 }
}
```

То же содержимое, указанное в качестве структуры каталогов, когда мы настраиваем хранилище ShadowNoFileLock

```
tree standalone/data/tx-object-store/
standalone/data/tx-object-store/
└── ShadowNoFileLockStore
   └── defaultStore
     ├── EISNAME
     │ └── 0_ffffc0a80065_-22769d16_60c87436_14
     └── StateManager
       └── BasicAction
         └── TwoPhaseCoordinator
           └── AtomicAction
             └── 0_ffffc0a80065_-22769d16_60c87436_1a
```

**log-store transactions and participant operations** - раздел транзакции и ресурсы участников содержит несколько операций, которые можно использовать для работы с содержимым хранилища объектов:

– «delete» - удаляет запись транзакции из хранилища объектов и выполняет вызов «XAResource.forget» для всех участников;

– «refresh» - перезагружает информацию об участнике из хранилища объектов «Narayana» и обновляет информацию из хранилища объектов в «model»;

– «recover» - эта операция изменяет статус участника на «PREPARED». В основном это полезно для эвристических записей участников, поскольку эвристическое состояние («HEURISTIC») пропускается при обработке восстановления периода.

Переключение «HEURISTIC» на «PREPARED» означает, что при периодическом восстановлении будет предпринята попытка завершить запись.

Операции в структуре транзакций *log*-хранилища:

```
# delete of the transaction that subsequently deletes all participants
/subsystem=transactions/log-store=log-
store/transactions=0\:ffffc0a80065\:-22769d16\:60c87436\:1a:delete
# delete of the particular participant
/subsystem=transactions/log-store=log-
store/transactions=0\:ffffc0a80065\:-
22769d16\:60c87436\:1a/participants=1:delete
# refresh and recover
/subsystem=transactions/log-store=log-
store/transactions=0\:ffffc0a80065\:-
22769d16\:60c87436\:1a/participants=1:refresh
/subsystem=transactions/log-store=log-
store/transactions=0\:ffffc0a80065\:-
22769d16\:60c87436\:1a/participants=1:recover
```

**client**

Конфигурация, связанная с клиентом транзакций WildBoss Pro.

Пример конфигурации XML для клиента:

```
<subsystem xmlns="urn:jboss:domain:transactions:6.0">
   ...
   <client stale-transaction-time="600"/>
   ...
</subsystem>
```

## 7.10 Конфигурация подсистемы метрик

**Внимание:** данная подсистема предоставляет только базовые показатели из модели управления WildBoss Pro и JVM MBeans.

**Внимание:** поддержка показателей микропрофилей обеспечивается подсистемой «microprofile-metrics-smallrye».

### 7.10.1 Расширение

Расширение «org.WildBoss Pro.extension.metrics» включено во все автономные конфигурации, включенные в дистрибутив WildBoss Pro, а также в уровень «metrics».

Также можно добавить расширение в конфигурацию без него, либо добавив элемент <extension module="org.WildBoss Pro.extension.metrics"/> в «xml», либо используя следующую операцию CLI:

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.metrics:add
```

### 7.10.2 Модель управления

Ресурс «/subsystem=metrics» определяет три атрибута:

– «security-enabled» - логическое значение, указывающее, требуется ли аутентификация для доступа к конечной точке HTTP metrics (описано ниже). По умолчанию это значение равно «true». В автономных конфигурациях явно устанавливается значение «false», чтобы разрешить доступ к конечным точкам HTTP без проверки подлинности;

– «exposed-subsystems» - список строк, соответствующих названиям подсистем, которые предоставляют свои метрики в конечных точках HTTP metrics. По умолчанию он не определен (подсистема не будет предоставлять метрики. Специальный подстановочный знак ««может использоваться для предоставления метрик из всех подсистем. В автономной конфигурации этому атрибуту присваивается значение»»;

– «prefix» - строка, добавляемая к метрикам WildBoss Pro, которые предоставляются конечной точкой HTTP «/metrics» в формате вывода «Prometheus».

### 7.10.3 Конечная точка HTTP

Конечная точка Metric HTTP доступна через интерфейс управления HTTP WildBoss Pro http://localhost:9990/metrics.

Защищенный доступ к конечной точке HTTP контролируется атрибутом «security-enabled» ресурса «/subsystem=metrics». Если для него установлено значение «true», HTTP-клиент должен пройти проверку подлинности.

Если безопасность отключена, конечная точка HTTP возвращает ответ 200 OK:

```
$ curl -v http://localhost:9990/metrics
< HTTP/1.1 200 OK
...
# HELP base_classloader_total_loaded_class_count Displays the total
number of classes that have been loaded since the Java virtual machine
has started execution
.
# TYPE base_classloader_total_loaded_class_count counter
base_classloader_total_loaded_class_count 10822.0
...
```

Если безопасность включена, HTTP-клиент должен передать учетные данные, соответствующие управляющему пользователю, созданному с помощью скрипта добавления пользователя. Например:

```
$ curl -v --digest -u myadminuser:myadminpassword
http://localhost:9990/metrics < HTTP/1.1 200 OK
...
# HELP base_classloader_total_loaded_class_count Displays the total
number of classes that have been loaded since the Java virtual machine
has started execution
.
# TYPE base_classloader_total_loaded_class_count counter
base_classloader_total_loaded_class_count 10822.0
...
```

Если аутентификация завершится неудачей, сервер выдаст ответ «401 NOT AUTHORIZED».

### 7.10.4 Открытые показатели

Конечная точка HTTP предоставляет следующие показатели:

– базовые показатели - показатели из JVM MBeans (считываются из их JMX MBeans);

– показатели поставщика - показатели WildBoss Pro из подсистемы модели управления и поддеревьев развертывания.

Конечная точка HTTP предоставляет метрики только в формате «Prometheus».

#### 7.10.4.1 Описание показателей WildBoss Pro

Названия метрик WildBoss Pro основаны на подсистеме, которая их предоставляет, а также на названии атрибута из модели управления. Перед их именем также может быть добавлен префикс (указанный на ресурсе «/subsystem=metrics»). Остальная информация хранится с помощью меток.

Например, «Undertow» предоставляет атрибут метрики «request-count» для каждого сервлета при развертывании приложения. Этот атрибут будет предоставлен «Prometheus» с именем «WildBoss Pro_undertow_request_count». Другая информация, такая как название сервлета, добавляется к меткам показателей.

Программа быстрого запуска [helloworld-rs] (https://github.com/WildBoss Pro/quickstart/tree/master/helloworld-rs) создает приложение «Jakarta RESTful Web Services», которое может быть развернуто в WildBoss Pro. Для него будет отображена соответствующая метрика с названием и метками:

```
WildBoss
Pro_undertow_request_count_total{deployment="helloworldrs.war",servlet="org.
jboss.as.quickstarts.rshelloworld.JAXActivator",subdeployment="hellowo rld-
rs.war"}
```

**Внимание:** некоторые подсистемы (например, «undertow» или «messaging-activemq») по умолчанию не включают статистику, поскольку это влияет на производительность и использование памяти. В этих подсистемах есть атрибут, поддерживающий статистику, для которого необходимо установить значение «true». Для удобства в автономной конфигурации WildBoss Pro предусмотрены выражения для включения статистики путем установки системного свойства – «DWildBoss Pro.statistics-enabled=true», чтобы включить статистику по подсистемам, предоставляемым конфигурацией.

## 7.11 Конфигурация подсистемы OpenTelemetry

### 7.11.1 Расширение

Это расширение не входит ни в одну из автономных конфигураций, включенных в дистрибутив WildBoss Pro. Для включения администратор должен выполнить следующие команды интерфейса командной строки:

```
$ jboss-cli.sh -c "/extension=org.WildBoss
Pro.extension.opentelemetry:add()"
$ jboss-cli.sh -c "/subsystem=opentelemetry:add()"
```

### 7.11.2 Конфигурация

Системные администраторы могут настроить ряд аспектов «OpenTelemetry»: экспортер, процессор «span» и выборочный модуль.

#### 7.11.2.1 Экспортер

Экспортер может быть выбран и сконфигурирован с помощью дочернего элемента «exporter», который поддерживает эти атрибуты:

– «exporter»: WildBoss Pro поддерживает двух разных экспортеров;

– «jaeger»: экспортер по умолчанию;

– «otlp»: протокол открытой телеметрии;

– «endpoint»: URL-адрес, по которому «OpenTelemetry» будет отправлять трассировки. По умолчанию используется конечная точка «Jaeger» на основе «gRPC», «http://localhost:14250».

#### 7.11.2.2 Процессор «Span»

Процесс «span» настраивается с помощью элемента «span-processor», который поддерживает следующие атрибуты:

– «type»: тип используемого процессора «span»:

– «batch»: процессор по умолчанию, который отправляет трассировки пакетами, как это настроено с помощью остальных атрибутов;

– «simple»: трассировки передаются экспортеру по мере их завершения;

– «batch-delay»: время ожидания в миллисекундах перед публикацией трассировок (по умолчанию: 5000);

– «max-queue-size»: максимальный размер очереди перед удалением трассировок (по умолчанию: 2048);

– «max-export-batch-size»: максимальное количество трассировок, публикуемых в каждом пакете, которое должно быть меньше или равно «максимальному размеру очереди» (по умолчанию: 512);

– «export-timeout»: максимальное количество времени в миллисекундах, которое требуется для завершения экспорта перед отменой (по умолчанию: 30000).

#### 7.11.2.3 Устройство для отбора проб

Сэмплер настраивается с помощью элемента «sampler»:

– «type»: тип используемого пробоотборника;

– «on»: всегда включен (все трассы записываются);

– «off»: всегда выключен (следы не записываются);

– «ratio»: возвращает соотношение трасс (например, 1 трасса из 10000);

– «ratio»: значение, используемое для настройки выборки коэффициента, которое должно быть в пределах [0,0, 1,0]. Например, если требуется экспортировать 1 трассировку из 10 000, это значение будет равно 0,0001.

#### 7.11.2.4 Пример конфигурации

Следующий XML-файл является примером полной конфигурации, включая значения по умолчанию (WildBoss Pro обычно не сохраняет значения по умолчанию, поэтому то, что вы видите в файле конфигурации, может выглядеть по-другому):

```
<subsystem xmlns="urn:WildBoss Pro:opentelemetry:1.0"
    service-name="example">
  <exporter
    type="jaeger"
    endpoint="http://localhost:14250"/>
  <span-processor
    type="batch"
    batch-delay="4500"
    max-queue-size="128"
    max-export-batch-size="512"
    export-timeout="45"/>
  <sampler
    type="on"/>
```

```
</subsystem>
```

### 7.11.3 Использование приложения

Все входящие запросы «REST» автоматически отслеживаются, поэтому в пользовательских приложениях не требуется выполнять никакой работы. Если получен запрос «REST» и присутствует заголовок распространения контекста «OpenTelemetry» (traceparent), запрос будет автоматически отслеживаться как часть контекста удаленной трассировки.

Аналогично, все клиентские вызовы Jakarta REST будут содержать контекст трассировки, добавленный к заголовкам исходящих запросов , чтобы запросы к внешним приложениям можно было правильно отслеживать (при условии, что удаленная система должным образом обрабатывает распространение контекста трассировки «OpenTelemetry»).Если клиентский вызов REST выполняется для другого приложения на локальном сервере WildBoss Pro или удаленном сервере той же версии или более поздней, контекст трассировки будет распространяться автоматически, как описано выше.

Хотя во многих случаях автоматической трассировки может быть достаточно, часто бывает желательно, чтобы трассировка выполнялась во всем пользовательском приложении. Чтобы поддержать это, WildBoss Pro предоставляет доступ к экземплярам «io.opentelemetry.api.OpenTelemetry» и «io.opentelemetry.api.trace.Tracer» с помощью внедрения CDI. Пользовательское приложение затем может создавать произвольные интервалы как часть трассировки, управляемой сервером:

```
@Path("/myEndpoint")
public class MyEndpoint {
   @Inject
   private Tracer tracer;
   @GET
   public Response doSomeWork() {
      final Span span = tracer.spanBuilder("Doing some work")
         .startSpan();
      span.makeCurrent();
      doSomeMoreWork();
      span.addEvent("Make request to external system.");
      makeExternalRequest();
      span.addEvent("All the work is done.");
      span.end();
      return Response.ok().build();
}
```

### 7.11.4 Взаимодействие с открытым профилем микропрофиля

Если у вас включена подсистема открытой трассировки микропрофилей (например, через конфигурацию микропрофиля), вы можете увидеть повторяющиеся трассировки, экспортированные, например, для вызовов JAX-RS. Это связано с тем, что как «OpenTracing», так и «OpenTelemetry» поддерживают трассировку в конечных точках, и обе подсистемы настроены на экспорт трассировок в сборщик, такой как Jaeger. Скорее всего, это нежелательно, поэтому, если вы хотите использовать «OpenTelemetry» для отслеживания, рекомендуется удалить подсистему «OpenTracing»:

```
$ jboss-cli.sh -c "/subsystem=microprofile-opentracing-
smallrye:remove()"
$ jboss-cli.sh -c "/extension=org.WildBoss
Pro.extension.microprofile.opentracingsmallrye:remove()"
```

Однако, если вы оставите обе подсистемы включенными, WildBoss Pro продолжит функционировать в обычном режиме. Единственное отличие, которое вы увидите, - это упомянутый выше дублирующий экспорт трассировки.

### 7.11.5 Ссылка на компонент

Поддержка «OpenTelemetry» осуществляется через проект «OpenTelemetry».

## 7.12 Конфигурация подсистемы работоспособности

**Внимание:** эта подсистема предоставляет только проверки работоспособности для среды выполнения WildBoss Pro. Поддержка работоспособности микропрофилей обеспечивается подсистемой «microprofile-health-smallrye».

### 7.12.1 Расширение

Это расширение «org.WildBoss Pro.extension.health» включено во все автономные конфигурации , входящие в дистрибутив WildBoss Pro, а также в уровень «health». Вы также можете добавить расширение в конфигурацию без него, либо добавив элемент <extension module="org.WildBoss Pro.extension.health"/> в «xml», либо используя следующую операцию CLI:

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.health:add
```

### 7.12.2 Модель управления

Параметр «/subsystem=health» ресурс работоспособности определяет один атрибут:

– безопасность включена - логическое значение, указывающее, требуется ли проверка подлинности для доступа к HTTP конечная точка работоспособности (описана ниже). По умолчанию это значение равно «true». В автономных конфигурациях для него явно задано значение «false», чтобы разрешить доступ к конечным точкам HTTP без проверки подлинности.

### 7.12.3 Конечная точка HTTP

Конечная точка работоспособности HTTP доступна в интерфейсе управления HTTP WildBoss Pro «http://localhost:9990/health».

Подсистема работоспособности регистрирует три конечные точки HTTP:

– «/health» для проверки работоспособности и готовности сервера приложений;

– «/health/live» для проверки работоспособности сервера приложений;

– «/health/ready» для проверки готовности сервера приложений;

– «/health/started» чтобы протестировать запуск сервера приложений.

Конечные точки работоспособности HTTP доступны в интерфейсе управления HTTP WildBoss Pro (например, http://localhost:9990/health).

Если сервер приложений исправен, он вернет ответ 200 OK:

```
$ curl -v http://localhost:9990/health
< HTTP/1.1 200 OK
```

Если сервер приложений неисправен, он возвращает сообщение «503 Service Unavailable»:

```
$ curl -v http://localhost:9990/health
< HTTP/1.1 503 Service Unavailable
```

#### 7.12.3.1 Защищенный доступ к конечным точкам HTTP

Защищенный доступ к конечной точке HTTP контролируется атрибутом «security-enabled». Если для него установлено значение «true», HTTP-клиент должен пройти проверку подлинности.

Если безопасность включена, HTTP-клиент должен передать учетные данные, соответствующие управляющему пользователю, созданному с помощью сценария добавления пользователя. Например:

```
$ curl -v --digest -u myadminuser:myadminpassword
http://localhost:9990/health
< HTTP/1.1 200 OK
```

Если аутентификация завершится неудачей, сервер выдаст ответ «401 NOT AUTHORIZED».

**Внимание:** HTTP-ответ содержит дополнительную информацию с индивидуальными результатами для каждого теста, который определил работоспособность. Это только информация, и код HTTP-ответа является единственными релевантными данными для определения работоспособности сервера приложений.

#### 7.12.3.2 Процедуры сервера по умолчанию

WildBoss Pro предоставляет некоторые процедуры готовности, которые проверяются, чтобы определить, готов ли сервер приложений к обслуживанию запросов:

– «boot-errors» проверяет, не было ли ошибок во время загрузки сервера;

– «deployments-status» проверяет, что все развертывания были выполнены без ошибок;

– «server-state» проверяет, запущено ли состояние сервера.

### 7.13 Конфигурация подсистемы настройки микропрофиля

Поддержка настройки микропрофилей обеспечивается подсистемой «microprofile-config-smallrye».

#### 7.13.1 Требуемое расширение

Требуемое расширение входит в состав стандартных конфигураций, включенных в дистрибутив WildBoss Pro.

Вы также можете добавить расширение в конфигурацию без него, либо добавив элемент <extension module="org.WildBoss Pro.extension.microprofile.config-smallrye"/> в «xml», либо используя следующую операцию CLI:

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.microprofile.configsmallrye:add
```

#### 7.13.2 Поддерживаемые источники конфигурации

В дополнение к конфигурационным источникам по умолчанию, указанным в спецификации конфигурации MicroProfile (переменные среды, системные свойства и файл «META-INF/microprofile-config.properties»), «microprofile-config-smallrye» предоставляет дополнительные типы конфигурационных источников.

#### 7.13.2.1 Конфигурационный источник из свойств

Можно сохранять свойства непосредственно в источнике конфигурации в подсистеме, используя атрибут «properties» при добавлении источника конфигурации:

```
/subsystem=microprofile-config-smallrye/config-
source=props:add(properties={"prop1" =
```

В результате получается конфигурация XML:

```
<subsystem xmlns="urn:WildBoss Pro:microprofile-config-smallrye:1.0">
  <config-source name="props">
    <property name="prop1" value="foo"/>
    <property name="prop2" value="bar"/>
  </config-source>
</subsystem>
```

#### 7.13.2.2 Конфигурационный источник из каталога

Также можно считывать свойства из каталога, где каждый файл - это имя свойства, а содержимое файла - это значение свойства.

Например, давайте представим, что каталог «/etc/config/numbers-app/» содержит 2 файла:

– файл «num.size» содержит значение 5;

– файл «num.max» содержит значение 100.

Можно создать «config-source»` для доступа к этим свойствам с помощью операции:

```
/subsystem=microprofile-config-smallrye/config-source=fileprops:
add(dir={path=/etc/config/numbers-app})
```

В результате получается конфигурация XML:

```
<subsystem xmlns="urn:WildBoss Pro:microprofile-config-smallrye:2.0">
 <config-source name="file-props">
  <dir path="/etc/config/numbers-app"/>
 </config-source>
</subsystem>
```

При такой конфигурации любое приложение, развернутое в WildBoss Pro, может использовать свойства «num.size» и «num.max» , которые хранятся в каталоге:

```
@Inject
@ConfigProperty(name = "num.size")
int numSize; ①
@Inject
@ConfigProperty(name = "num.max")
int numMax; ②
```

① будет установлено значение 5

② будет установлено значение 100

**Внимание:** это соответствует макету, используемому OpenShift ConfigMaps. Значение «dir» соответствует пути монтирования в определении ConfigMap в OpenShift или Kubernetes.

Конфигурационные источники из корневого каталога.

Вы также можете указать на корневой каталог, изменив примеры в предыдущем разделе, чтобы включить значение «root=true» при их определении. Каталоги верхнего уровня в этом корневом каталоге становятся отдельными «configSource», считываемыми из каталога, аналогично тому, что мы видели ранее в разделе «configSource from Directory». Все каталоги, расположенные ниже каталогов верхнего уровня, игнорируются. Также игнорируются любые файлы в корневом каталоге; для настройки будут использоваться только файлы в каталогах верхнего уровня в корневом каталоге.

Это особенно полезно при работе в OpenShift, где такие конструкции, как ConfigMap и экземпляры ServiceBinding, отображаются в общем известном местоположении. Например, если в данном модуле приложений используются два экземпляра ConfigMap (для этого примера назовем их «map-a» и «map-b») , каждый из них будет сопоставлен в «/etc/config». Таким образом, будут каталоги «/etc/config/mapa» и «/etc/config/map-b».

В каждом из этих каталогов будут содержаться файлы, где имя файла является именем свойства, а содержимое файла - значением свойства, как мы видели ранее.

Таким образом, можно просто запустить следующую команду CLI, чтобы выбрать все эти дочерние каталоги в качестве configSource для каждого:

```
/subsystem=microprofile-config-smallrye/config-source=config-maproot:
add(dir={path=/etc/config, root=true})
```

В результате получается конфигурация XML:

```
<subsystem xmlns="urn:WildBoss Pro:microprofile-config-smallrye:2.0">
  <config-source name="config-map-root">
    <dir path="/etc/config" root="true"/>
  </config-source>
</subsystem>
```

Предполагая, что каталог «/etc/config» содержит каталоги «map-a» и «map-b», которые мы используем в этом примере, вышеописанное аналогично выполнению:

```
/subsystem=microprofile-config-smallrye/config-
source=fileprops:add(dir={path=/etc/config/map-a})
/subsystem=microprofile-config-smallrye/config-
source=fileprops:add(dir={path=/etc/config/map-b})
```

Указание корневого каталога, а не каждого отдельного каталога устраняет необходимость знать точные имена каждой записи в общем родительском каталоге (это особенно полезно в некоторых случаях). Сценарии OpenShift, в которых имена этих каталогов генерируются автоматически).

Следует избегать ситуации, когда две записи «configSource» с одним и тем же корнем содержат одно и то же свойство. Однако, чтобы сделать эту ситуацию детерминированной, каталоги, представляющие каждый из «configSource», сортируются по имени в соответствии со стандартными правилами сортировки Java перед выполнением поиска значений. Чтобы сделать это более конкретным, если следующие записи: «* /etc/config/mapa/» имя содержит «kabir * /etc/config/map-b/» имя содержит «jeff».

Поскольку после сортировки «map-a» будет предшествовать «map-b», в следующем сценарии «kabir» (исходящий из «map-a») будет введен для следующего поля имени пользователя:

```
@Inject
@ConfigProperty(name = "name")
String username;
```

Можно изменить сортировку по умолчанию, включив в каталог файл с именем «config_ordinal». Порядковый номер, указанный в этом файле, будет использоваться для значений конфигурации, поступающих из этого каталога. Основываясь на нашем предыдущем примере, если бы у нас было: «* /etc/config/map-a/config_ordinal» содержит 120 «* /etc/config/map-b/config_ordinal» содержит 140.

Поскольку теперь «map-b» имеет более высокий порядковый номер (140), чем «map-a» (120), мы вместо этого введем значение «jeff» в предыдущее поле «username».

Если в каталоге верхнего уровня под корневым каталогом нет файла «config_ordinal», для этого каталога будет использоваться порядковый номер, использованный при указании источника конфигурации.

#### 7.13.2.3 ConfigSource из класса

Можно создать определенный тип реализации «configSource», создав ресурс «config-source» с атрибутом «class».

Например, можно предоставить реализацию «org.eclipse.microprofile.config.spi.configSource» с именем «org.example.MyConfigSource», которая предоставляется модулем JBoss с именем «org.example»:

```
<subsystem xmlns="urn:WildBoss Pro:microprofile-config-smallrye:2.0">
  <config-source name="my-config-source">
    <class name="org.example.MyConfigSource" module="org.example"/>
  </config-source>
</subsystem>
```

Все свойства из этого источника конфигурации будут доступны для любого развертывания WildBoss Pro.

#### 7.13.2.4 ConfigSourceProvider из класса

Можно создать определенный тип реализации «ConfigSourceProvider», создав ресурс «config-sourceprovider» с атрибутом «class».

Например, вы можете предоставить реализацию «org.eclipse.microprofile.config.spi.ConfigSourceProvider» с именем «org.example.MyConfigSourceProvider» и предоставляется модулем JBoss с именем «org.example»:

```
/subsystem=microprofile-config-smallrye/config-source-provider=my-
config-
sourceprovider:add(class={name=org.example.MyConfigSourceProvider,
module=org.example})
```

В результате получается конфигурация XML:

```
<subsystem xmlns="urn:WildBoss Pro:microprofile-config-smallrye:2.0">
  <config-source-provider name="my-config-source-provider">
    <class name="org.example.MyConfigSourceProvider"
    module="org.example"/>
  </config-source-provider>
</subsystem>
```

Все свойства из «configSource», предоставляемые этим «ConfigSourceProvider", будут доступны для любого развертывания WildBoss Pro.

#### 7.13.3 Развертывание

В приложениях, развернутых в WildBoss Pro, должны быть включены контексты Jakarta и внедрение зависимостей (например, с помощью «META-INF/beans.xml» или с помощью контекстов Jakarta и внедрения зависимостей Bean-аннотации), чтобы иметь возможность использовать конфигурацию микропрофиля в своем коде.

#### 7.13.4 Ссылка на компонент

Реализация конфигурации микропрофиля обеспечивается проектом SmallRye Config.

```
  • MicroProfile Config
  • SmallRye Config
```

### 7.14 Конфигурация подсистемы работоспособности микропрофиля

Поддержка работоспособности микропрофиля обеспечивается подсистемой «microprofile-health-smallrye».

#### 7.14.1 Требуемое расширение

Это расширение включено в конфигурации standalone-microprofile, входящие в дистрибутив WildBoss Pro.

Также можно добавить расширение в конфигурацию без него, добавив <extension module="org.WildBoss Pro.extension.microprofile.health-smallrye"/> в «xml» или с помощью следующей операции CLI:

```
[standalone@localhost:9990 /]/extension=org.WildBoss
Pro.extension.microprofile.healthsmallrye:add
```

Это зависит от базового расширения работоспособности «org.WildBoss Pro.extension», которое должно быть установлено.

#### 7.14.2 Управленческие операции

Работоспособность сервера приложений можно проверить, выполнив 3 различные операции:

– «check» чтобы проверить как работоспособность, так и готовность среды выполнения;

– «check-live» чтобы проверить только работоспособность среды выполнения;

– «check-ready» чтобы проверить только готовность среды выполнения;

– «check-started» чтобы проверить только запуск среды выполнения.

```
[standalone@localhost:9990 /] /subsystem=microprofile-health-
smallrye:check
{
  "outcome" => "success", ①
  "result" => {
    "status" => "UP", ②
    "checks" => [
    {
      "name" => "server-state",
      "status" => "UP",
      "data" => {"value" => "running"}
    },
    {
      "name" => "empty-startup-checks",
      "status" => "UP"
    },
    {
      "name" => "empty-readiness-checks",
      "status" => "UP"
    },
    {
      "name" => "boot-errors",
      "status" => "UP"
    },
    {
      "name" => "empty-liveness-checks",
      "status" => "UP"
    },
    {
      "name" => "deployments-status",
      "status" => "UP"
    }
    ]
  }
}
```

① этот результат означает, что операция по управлению прошла успешно

② этот статус соответствует проверке работоспособности, повышается, если сервер приложений исправен, и понижается в противном случае

### 7.14.3 Конечные точки HTTP

Спецификации проверки работоспособности микропрофиля определяют три конечные точки HTTP:

– «/health» для проверки работоспособности и готовности сервера приложений;

– «/health/live» чтобы проверить работоспособность сервера приложений;

– «/health/ready» для проверки готовности сервера приложений;

– «/health/started» чтобы протестировать запуск сервера приложений.

Конечные точки работоспособности HTTP доступны в интерфейсе управления HTTP WildBoss Pro (например, «http://localhost:9990/health»).

Если сервер приложений исправен, он вернет ответ 200 OK:

```
$ curl -v http://localhost:9990/health
< HTTP/1.1 200 OK
...
{"status":"UP","checks":[{"name":"serverstate","
status":"UP","data":{"value":"running"}},{"name":"empty-
startupchecks","
status":"UP"},{"name":"empty-readiness-
checks","status":"UP"},{"name":"booterrors","
status":"UP"},{"name":"empty-livenesschecks","
status":"UP"},{"name":"deployments-status","status":"UP"}]}
```

Если сервер приложений неисправен, он возвращает сообщение «503 Service Unavailable».

```
$ curl -v http://localhost:9990/health
< HTTP/1.1 503 Service Unavailable
...
{"outcome":"DOWN","checks":[{"name":"myFailingProbe","state":"DOWN","da
ta":{"foo":"bar
"}}]}
```

#### 7.14.3.1 Защищенный доступ к конечным точкам HTTP

Защищенный доступ к конечной точке HTTP контролируется атрибутом «security-enabled» ресурса «/subsystem=microprofile-health-smallrye», который поддерживает безопасность. Значение этого атрибута переопределяет значение атрибута с поддержкой безопасности ресурса «/subsystem=health» (описано в руководстве по настройке подсистемы работоспособности). Если для него установлено значение «true», HTTP-клиент должен пройти проверку подлинности.

Если безопасность включена, HTTP-клиент должен передать учетные данные, соответствующие управляющему пользователю, созданному с помощью скрипта добавления пользователя. Например:

```
$ curl -v --digest -u myadminuser:myadminpassword
http://localhost:9990/health
< HTTP/1.1 200 OK
...
{"status":"UP","checks":[{"name":"empty-livenesschecks","
status":"UP"},{"name":"serverstate","
status":"UP","data":{"value":"running"}},{"name":"booterrors","
status":"UP"},{"name":"deployments-
status","status":"UP"},{"name":"emptyreadiness-
checks","status":"UP"}]}
```

Если аутентификация завершится неудачей, сервер выдаст ответ «401 NOT AUTHORIZED».

#### 7.14.3.2 Процедуры сервера по умолчанию

WildBoss Pro предоставляет некоторые процедуры готовности, которые проверяются, чтобы определить, готов ли сервер приложений к обслуживанию запросов:

– «boot-errors» проверяет, не было ли ошибок во время загрузки сервера;

– «deployments-status» проверяет, что все развертывания были выполнены без ошибок;

– «server-state» проверяет, запущено ли состояние сервера;

– «empty-readiness-checks» определяет статус, когда на сервере не развернуты процедуры проверки готовности. Результат этой процедуры определяется атрибутом «empty-readinesschecks- status». Если атрибут имеет значение «UP» (по умолчанию), сервер может быть готов, когда в развертываниях нет проверок готовности. Установка атрибута состояния пустой проверки готовности на значение «DOWN» приведет к сбою этой процедуры, когда в развертываниях нет проверок готовности.

Если в развертывании не предусмотрены какие-либо проверки готовности, WildBoss Pro автоматически добавит одну проверку для каждого развертывания (с именем «ready-<deployment name>»), которая всегда возвращает «UP».

**Внимание:** это позволяет приложениям, которые не предоставляют проверки готовности, по-прежнему сообщать облачным контейнерам, когда они готовы обслуживать запросы. Установка значения «emptyreadiness- checks-status» в значение «DOWN» гарантирует, что сервер не будет готов до тех пор, пока приложение не будет развернуто. В это время будет добавлен параметр «ready-<deployment name>» (который возвращает значение «UP»), и процедура проверки пустой готовности больше не будет проверяться, поскольку теперь процедура проверки готовности предоставляется либо развертыванием, либо сервером.

WildBoss Pro также предоставляет процедуру проверки работоспособности, которая проверяется, чтобы определить, работает ли сервер приложений:

– «empty-liveness-checks» определяет статус, когда на сервере не развернуты процедуры проверки работоспособности. Результат этой процедуры определяется атрибутом «empty-livenesschecks- status». Если атрибут установлен (по умолчанию), сервер может быть подключен к сети, даже если в развертываниях нет проверок работоспособности. Установка атрибута состояния пустой проверки работоспособности на значение «DOWN» приведет к сбою этой процедуры, если в развертываниях нет проверок работоспособности.

WildBoss Pro также предоставляет аналогичную процедуру для проверки запуска:

– «empty-startup-checks» определяет статус, когда на сервере не развернуты процедуры проверки запуска. Результат этой процедуры определяется атрибутом «empty-startupchecks-status». Если для атрибута установлено значение UP (по умолчанию), сервер может быть готов, даже если в развертываниях не выполняются проверки состояния запуска. Установка значения «DOWN» для атрибута состояния пустых проверок запуска приведет к сбою этой процедуры, если в развертываниях не выполняются проверки готовности.

Если в развертывании не предусмотрены какие-либо проверки при запуске, WildBoss Pro автоматически добавит одну проверку для каждого развертывания (с именем «started-<deployment name>»), которая всегда возвращает «UP».

**Внимание:** это позволяет приложениям, в которых не предусмотрены проверки при запуске, по-прежнему сообщать облачным контейнерам о своем запуске, чтобы они продолжали запуск контейнера. Установка значения «неактивный статус проверки при запуске» означает, что сервер не будет готов до тех пор, пока приложение не будет развернуто. В это время будет добавлен параметр «started-<deployment name>» (который возвращает значение «UP»), и процедура проверки пустого запуска больше не будет проверяться, поскольку теперь процедура проверки запуска предоставляется либо развертыванием, либо сервером.

#### 7.14.3.3 Отключение серверных процедур по умолчанию

Все эти серверные процедуры можно отключить, используя свойство конфигурации микропрофиля «mp.health.disable-default-procedures».

**Внимание:** Свойство конфигурации микропрофиля «mp.health.disable-default-procedures» считывается в 2 разных момента времени:

1) при запуске сервера, чтобы определить, следует ли отключить или включить его серверные процедуры. Его можно задать с помощью системного свойства mp.health.disable-defaultprocedures или переменной среды MP_HEALTH_DISABLE_DEFAULT_PROCEDURES. Установка этого свойства при развертывании в это время игнорируется;

2) когда приложение будет развернуто, чтобы определить, следует ли WildBoss Pro добавить проверку готовности, если при развертывании она не выполняется. В то время установка этого свойства в файле «microprofile-config.properties» при развертывании будет приниматься во внимание. (с обычными правилами приоритета для свойств конфигурации микропрофиля).

Когда параметру «mp.health.disable-default-procedures» присвоено значение «true», сервер не будет возвращать ни одну из своих проверок работоспособности в ответах, которые также включают в себя пустые настраиваемые проверки по умолчанию, включенные перед обработкой развертываний, а именно пустые проверки готовности, пустые проверки запуска и пустые проверки работоспособности- чеки. Это означает, что сервер может преждевременно выдать неверный ответ «UP», особенно на вызовы «startup» и «readiness» до того, как будет обработано пользовательское развертывание. По этой причине спецификация работоспособности микропрофиля определяет два свойства конфигурации микропрофиля, которые определяют возвращаемый ответ, когда сервер все еще обрабатывает развертывания, т.е. возвращает пустой ответ о работоспособности:

– «mp.health.default.readiness.empty.response» (по умолчанию «DOWN»), который указывает пустой ответ о готовности. Этот ответ будет изменен на «UP», как только пользовательское развертывание будет обработано, даже если оно не содержит никаких проверок готовности. В противном случае он будет изменен на статус, установленный пользовательскими проверками готовности;

– «mp.health.default.startup.empty.response» (по умолчанию «DOWN»), который указывает пустой ответ при запуске. Этот ответ будет изменен на «UP», как только пользовательское развертывание будет обработано, даже если оно не содержит никаких проверок при запуске. В противном случае он будет изменен на статус, установленный пользовательскими проверками при запуске.

### 7.14.4 Ссылка на компонент

Внедрение микропрофиля здравоохранения осуществляется в рамках проекта «SmallRye Health».

- MicroProfile Health
- SmallRye Health

## 7.15 Конфигурация подсистемы JWT с микропрофилем

Поддержка RBAC для микропрофилей JWT обеспечивается подсистемой «microprofile-jwt-smallrye».

Спецификация JWT для микропрофилей описывает, как может быть выполнена аутентификация с использованием криптографически подписанных токенов JWT, а также содержимое токена, которое будет использоваться для установления повторной идентификации, не полагаясь на доступ к внешним хранилищам идентификационных данных, таким как базы данных или серверы каталогов.

### 7.15.1 Подсистема

Интеграция MicroProfile JWT обеспечивается подсистемой «microprofile-jwt-smallrye» и включена в конфигурацию по умолчанию, если подсистема отсутствует, ее можно добавить, используя следующие Команды CLI.

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.microprofile.jwtsmallrye:add
[standalone@localhost:9990 /] /subsystem=microprofile-jwt-smallrye:add
```

На этом этапе сервер необходимо будет перезагрузить, чтобы активировать изменение.

### 7.15.2 Конфигурация

Подсистема «microprofile-jwt-smallrye» не содержит настраиваемых атрибутов или ресурсов, однако ее наличие требуется для определения того, использует ли развертывание механизм аутентификации MP-JWT , и для активации поддержки JWT, использующей проект SmallRye JWT.

#### 7.15.2.1 Активация

Подсистема проверит все развертывания, чтобы определить, требуется ли механизм MP-JWT для каких-либо веб-компонентов, и, если этот параметр установлен, активирует интеграцию и механизм аутентификации.

Классы в развертывании будут проверены, чтобы определить, есть ли класс, который расширяет «javax.ws.rs.core.Application», помеченное как «org.eclipse.microprofile.auth.LoginConfig», чтобы указать метод аутентификации. Кроме того, будет проверен метод аутентификации, содержащийся в развертываниях «web.xml».

Если конфигурация аутентификации определена в аннотации «@LoginConfig» и в дескрипторе развертывания web.xml, то приоритет отдается содержимому «web.xml». Если после оценки развертывания результирующим методом аутентификации будет MP-JWT, то эта интеграция будет активирована, во всех остальных случаях активация не произойдет, и развертывание продолжится в обычном режиме.

#### 7.15.2.2 Конфигурация микропрофиля

Для индивидуального развертывания конфигурация в отношении MicroProfile JWT может быть предоставлена с использованием свойств конфигурации MicroProfile, многие из которых определены в спецификации MicroProfile JWT, однако SmallRye JWT также поддерживает некоторые дополнительные свойства.

Таблица 6 - Стандартные свойства

| Свойство | Значение по умолчанию | Описание |
|---|---|---|
| mp.jwt.verify.publickey | NONE | Строковое представление открытого ключа, закодированного с использованием одного из поддерживаемых форматов. Не следует устанавливать одновременно с mp.jwt.verify.publickey |
| mp.jwt.verify.publickey.location | NONE | Расположение открытого ключа может быть относительным путем или URL-адресом. Не следует указывать одновременно с mp.jwt.verify.publickey.location |
| mp.jwt.verify.issuer | NONE | Ожидаемое значение любого утверждения «iss» для любого проверяемого токена JWT |

Минимальный «microprofile-config.properties» может выглядеть следующим образом

```
mp.jwt.verify.publickey.location=META-INF/public.pem
mp.jwt.verify.issuer=quickstart-jwt-issuer
```

Недоступные опции.

В настоящее время существует несколько ограничений, связанных с поддержкой JWK, которые мы стремимся устранить:

– если JWKS встроен с использованием свойства «mp.jwt.verify.publickey», то будет использоваться только первый ключ из набора, а остальные будут проигнорированы;

– кодирование JWKS с использованием Base64 в настоящее время не поддерживается.

В обоих случаях вместо этого можно ссылаться на JWKS с открытым текстом, используя «mp.jwt.verify.publickey.location» свойство конфигурации.

Поддержка JWKS-ключей в кодировке Base64 и встроенных JWKS-ключей в свойстве «mp.jwt.verify.publickey» будет дополнительно изучена, и будет либо добавлена поддержка, либо в спецификации будет добавлено дополнение для удаления этих параметров.

Специфические свойства SmallRye JWT допускают множество настроек, не предусмотренных спецификацией, однако, поскольку они не определены спецификацией, они могут быть изменены.

Таблица 7 - Небольшие свойства JWT

| **Свойство** | **Значение по умолчанию** | **Описание** |
|---|---|---|
| smallrye.jwt.token.header | Authorization | HTTP-заголовок для извлечения токена из |
| smallrye.jwt.token.cookie | NONE | Имя файла «cookie» для извлечения маркера, только применяется, если «smallrye.jwt.token.header» имеет значение файла «cookie» |
| smallrye.jwt.token.kid | NONE | Ожидаемое значение «kid» для любого проверяемого токена |
| smallrye.jwt.require.named-principal | false | При значении «true» требуется, чтобы любой проверяемый токен содержал хотя бы одно из следующих значений: «sub», «upn», «preferred_user_name» |
| smallrye.jwt.claims.sub | NONE | Значение по умолчанию, используемое для любого вложенного утверждения, если оно опущено во входящем токене |
| smallrye.jwt.path.sub | NONE | Путь к формуле, содержащей вспомогательную формулу, позволяет вложить ее в альтернативную формулу |
| smallrye.jwt.claims.groups | NONE | Значение по умолчанию для любого группового требования, если оно не указано во входящем токене |
| smallrye.jwt.path.groups | NONE | Путь к утверждению, содержащему утверждение «groups», позволяет вложить его в альтернативное утверждение |
| smallrye.jwt.groups-separator | {SPACE} | Где «smallrye.jwt.path.groups» ссылается на альтернативный путь - разделитель для разделения значения на отдельные группы |
| smallrye.jwt.expiration.grace | 60 | Льготный период в секундах токен будет принят после его истечения |

| | | |
|---|---|---|
| smallrye.jwt.jwks.refresh-interval | 60 | В обновлении inerval, где mp.jwt.verify.publickey.location указывает на местоположение по протоколу HTTPS |
| smallrye.jwt.whitelist.algorithms | RS256 | Список поддерживаемых алгоритмов |
| smallrye.jwt.verify.aud | NONE | Ожидаемые значения аудитории для токена |

### 7.15.3 Виртуальная безопасность

Для традиционных развертываний в WildBoss Pro, где требуется безопасность, во время развертывания будет определено доменное имя безопасности, которое, в свою очередь, будет сопоставлено для использования настроенных ресурсов либо в «elytron», либо в устаревших подсистемах безопасности.

Одной из основных причин использования MicroProfile JWT является возможность описать идентификатор с помощью входящего токена, не полагаясь на доступ к внешним ресурсам. По этой причине MicroProfile развертывания JWT не будут зависеть от управляемых ресурсов SecurityDomain, вместо этого будет создан виртуальный SecurityDomain будет создан и использоваться во время всего развертывания.

Поскольку развертывание полностью настроено в соответствии со свойствами конфигурации MicroProfile, за исключением наличия подсистемы «microprofile-jwt-smallrye», виртуальный домен безопасности означает, что для развертывания не требуется никакой другой управляемой конфигурации.

## 7.16 Конфигурация подсистемы показателей MicroProfile

Поддержка показателей MicroProfile обеспечивается подсистемой «microprofile-metrics-smallrye».

### 7.16.1 Требуемое расширение

Это расширение включено в конфигурации автономных MicroProfile, входящие в дистрибутив WildBoss Pro.

Вы также можете добавить расширение в конфигурацию без него, либо добавив элемент <extension module="org.WildBoss Pro.extension.microprofile.metrics-smallrye"/> в «xml», либо используя следующую операцию CLI:

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.microprofile.metricssmallrye:add
```

Это зависит от расширения базовых показателей «org.WildBoss Pro.extension.metrics», которое должно быть установлено.

### 7.16.2 Модель управления

Ресурс «/subsystem=microprofile-metrics-smallrye» определяет два атрибута:

– «exposed-subsystems» - список строк, соответствующих названиям подсистем, которые предоставляют свои метрики в конечных точках HTTP metrics. По умолчанию он не определен (подсистема не будет предоставлять метрики. Специальный подстановочный знак "" может использоваться для предоставления метрик из всех подсистем. В автономной конфигурации этому атрибуту присваивается значение "";

– «prefix» - строка, добавляемая к метрикам WildBoss Pro, которые предоставляются конечной точкой HTTP /metrics в формате вывода Prometheus.

### 7.16.3 Конечная точка HTTP

Конечная точка Metric HTTP доступна в интерфейсе управления HTTP WildBoss Pro «http://localhost:9990/metrics».

Защищенный доступ к конечной точке HTTP контролируется атрибутом «security-enabled» ресурса «/subsystem=microprofile-metrics-smallrye», который поддерживает безопасность. Значение этого атрибута переопределяет атрибут с поддержкой безопасности ресурса «/subsystem=metrics» (описанный в руководстве по настройке подсистемы Metrics).

При установке «microprofile.metrics-smallrye» эта конечная точка HTTP будет возвращать метрики в соответствии со спецификацией MicroProfile Metrics.

```
$ curl -v http://localhost:9990/metrics
< HTTP/1.1 200 OK
...
# HELP base:classloader_total_loaded_class_count Displays the total
number of classes
that have been loaded since the Java virtual machine has started
execution
.
# TYPE base:classloader_total_loaded_class_count counter
base:classloader_total_loaded_class_count 10822.0
...
```

### 7.16.4 Открытые показатели

Конечная точка HTTP предоставляет следующие показатели:

– базовые показатели - обязательные показатели, указанные в спецификации MicroProfile 1.1, отображаются в базовой области;

– показатели поставщика - показатели, зависящие от конкретного поставщика (например, для пулов памяти);

– показатели приложения - показатели из приложения и подсистем развертывания отображаются в области приложения.

Показатели WildBoss Pro (которые измеряют активность в подсистеме или развертывании приложений) доступны только на конечной точке «/metrics» в формате «Prometheus».

### 7.16.5 Ссылка на компонент

Реализация MicroProfile Metrics обеспечивается проектом SmallRye Metrics.

- MicroProfile Metrics
- SmallRye Metrics

## 7.17 Конфигурация подсистемы OpenAPI с MicroProfile

Спецификация OpenAPI определяет контракт для приложений JAX-RS таким же образом, как WSDL определяет контракт для устаревших веб-служб. Спецификация OpenAPI для микропрофилей определяет механизм создания документа OpenAPI версии 3 из приложения JAX-RS, а также API для настройки создания документа.

### 7.17.1 Подсистема

Поддержка OpenAPI для MicroProfile обеспечивается подсистемой «microprofile-openapi-smallrye». Эта подсистема включена в конфигурацию дистрибутива WildBoss Pro по умолчанию «standalone-microprofile.xml».

Вы также можете вручную добавить подсистему в любой профиль с помощью интерфейса командной строки:

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.microprofile.openapi-
smallrye:add()
 [standalone@localhost:9990 /] /subsystem=microprofile-openapi-
 smallrye:add()
```

### 7.17.2 Конфигурация

Подсистема «microprofile-openapi-smallrye» получает всю свою конфигурацию с помощью MicroProfile Конфигурация. Таким образом, сама подсистема не определяет никаких атрибутов.

В дополнение к стандартным свойствам конфигурации Open API, WildBoss Pro поддерживает дополнительные свойства конфигурации MicroProfile, преведенные в таблице 8

Таблица 8 - Дополнительные свойства конфигурации MicroProfile

| Свойство | Значение по умолчанию | Описание |
|---|---|---|
| mp.openapi.extensions.enabled | true | Включает/отключает регистрацию конечной точки OpenAPI. Многие пользователи захотят настроить этот параметр, чтобы выборочно включать/отключать OpenAPI в разных средах |
| mp.openapi.extensions.path | /openapi | Используется для настройки пути к конечной точке OpenAPI |
| mp.openapi.extensions.servers.relative | true | Указывает, являются ли автоматически сгенерированные серверные записи абсолютными или относительными к местоположению конечной точки OpenAPI. Если они абсолютны, WildBoss Pro сгенерирует серверные записи, содержащие протоколы, хосты и порты, по которым доступно данное развертывание |

Например, /META-INF/microprofile-config.properties:

```
mp.openapi.extensions.enabled=${microprofile.openapi.enabled}
mp.openapi.extensions.path=/swagger
mp.openapi.extensions.servers.relative=false
```

### 7.17.3 Конечная точка HTTP/S

Спецификация OpenAPI для MicroProfile определяет конечную точку HTTP, которая обслуживает документ OpenAPI 3.0, описывающий конечные точки REST для хоста. Конечная точка OpenAPI регистрируется с использованием настроенного пути (например, «http://localhost:8080/openapi»), локального по отношению к корневому каталогу хоста, связанного с данным развертыванием.

**Внимание:** в настоящее время конечная точка OpenAPI для данного виртуального хоста может документировать только одно развертывание JAX-RS. Чтобы использовать OpenAPI для нескольких развертываний JAX-RS, зарегистрированных с разными контекстными путями на одном и том же виртуальном хосте, для каждого развертывания должен использоваться отдельный путь к конечной точке.

По умолчанию конечная точка OpenAPI возвращает документ YAML. В качестве альтернативы документ JSON может быть запрошен с помощью заголовка Accept HTTP или параметра запроса «format».

Например:

```
$ curl -v http://localhost:8080/openapi?format=JSON
< HTTP/1.1 200 OK
...
{"openapi": "3.0.1" ... }
$ curl -v -H'Accept: application/json' http://localhost:8080/openapi
< HTTP/1.1 200 OK
...
{"openapi": "3.0.1" ... }
```

Если сервер/хост Undertow данного приложения определяет прослушиватель HTTPS, то документ OpenAPI также будет доступен по протоколу HTTPS, например «https://localhost:8443/openapi».

### 7.17.4 Ссылка на компонент

Реализация OpenAPI для микропрофилей обеспечивается проектом SmallRye OpenAPI.

- MicroProfile OpenAPI
- SmallRye OpenAPI

**Внимание:** ссылки в этом документе на Java API для веб-служб RESTful(JAX-RS) относятся к Веб-службам RESTful для Jakarta, если не указано иное.

## 7.18 Конфигурация подсистемы OpenTracing MicroProfile

Поддержка MicroProfile OpenTracing предоставляется в качестве функции технического просмотра подсистемой «microprofileopentracing- smallrye».

### 7.18.1 Требуемое расширение

Это расширение включено в стандартные конфигурации, входящие в дистрибутив WildBoss Pro.

Также можно добавить расширение в конфигурацию без него, добавив <extension module="org.WildBoss Pro.extension.microprofile.opentracing-smallrye"/> в «xml» или с помощью следующей операции CLI:

```
[standalone@localhost:9990 /]
/extension=org.WildBoss Pro.extension.microprofile.opentracing-
smallrye:add
{"outcome" => "success"}

[standalone@localhost:9990 /] /subsystem=microprofile-opentracing-
smallrye:add
{
   "outcome" => "success",
   "response-headers" => {
     "operation-requires-reload" => true,
     "process-state" => "reload-required"
   }
}
```

### 7.18.2 Поддерживаемые библиотеки инструментальных средств

Подсистема OpenTracing MicroProfile от WildBoss Pro реализует версию MicroProfile 1.3, которая включает поддержку отслеживания веб-служб RESTful Jakarta и контекстов Jakarta, а также внедрение зависимостей.

В настоящее время подсистема позволяет настраивать Java-клиент Jaeger. Затем можно указать, какой трассировщик использовать в вашем развертывании, указав параметр «smallrye.opentracing.tracer.configuration» в данном дескрипторе развертывания. Если это значение неверно, развертывание завершится ошибкой. Также можно настроить трассировку по умолчанию, используя атрибут «default-tracer» в конфигурации подсистемы.

Кроме того, развертываемые приложения могут предоставлять свои собственные средства трассировки с помощью средства TracerResolver. В этом случае средство трассировки по умолчанию использоваться не будет.

Для обеспечения совместимости, если в подсистеме не определен трассировщик и его невозможно разрешить, то из переменных среды будет создан экземпляр трассировщика Jaeger.

### 7.18.3 Настройка трассировщика Jaeger

Чтобы добавить новый экземпляр Jaeger tracer, можно использовать следующую операцию CLI:

```
[standalone@localhost:9990 /] /subsystem=microprofile-opentracing-
smallrye/jaegertracer=
my-tracer:add()
{"outcome" => "success"}
```

Это атрибуты конфигурации:

– «propagation» - поддерживаемые форматы распространения контекста трассировки:

а) «JAEGER» - формат распространения контекста трассировки Jaeger по умолчанию;

б) «B3» - формат распространения контекста трассировки Zipkin B3;

– «sampler-type» - тип пробоотборника, который будет использоваться в трассировщике. Необязательный. Допустимые значения: удаленный (по умолчанию), ограничивающий скорость, вероятностный, постоянный;

– «sampler-param» - значение с плавающей запятой, которое имеет смысл для правильного типа выборки;

– «sampler-manager-host-port» - порт менеджера по отбору проб, который может предоставить стратегию отбора проб для этой службы;

– «sender-binding» - привязка исходящего сокета для подключения к агенту;

– «sender-endpoint» - конечная точка, например «https://jaeger-collector:14268/api/traces»;

– «sender-user» - основное имя пользователя «Auth», которое будет добавлено в заголовки авторизации для запросов, отправляемых в конечную точку;

– «sender-auth-password» - базовый пароль для авторизации, который будет добавлен в заголовки авторизации для запросов, отправляемых в конечную точку;

– «sender-auth-token» - токен аутентификации, который будет добавлен в качестве "предъявителя" в заголовки авторизации для запросов, отправляемых в конечную точку;

– «reporter-log-spans» - логическое значение «to» указывает, должен ли репортер регистрировать промежутки;

– «reporter-flush-interval» - удаленный интервал очистки при составлении отчетов составляет миллисекунды;

– «reporter-max-queue-size» - максимальный размер очереди репортера;

– «tracer_id_128bit» - необходимо выбрать 128-разрядный идентификатор трассировки. По умолчанию используется 64-разрядный идентификатор;

– «tracer-tags» - список тегов уровня трассировки «name = value», разделенных запятыми, которые добавляются ко всем отчетным интервалам. Значение также может ссылаться на переменную среды, используя формат «${envVarName:default}», где значение «:default» является необязательным и определяет значение, которое будет использоваться, если переменная среды не может быть найдена.

По умолчанию имя службы, используемое в клиенте Jaeger, является производным от имени развертывания, которое обычно является именем файла «WAR».

Как определено в спецификации OpenTracing для MicroProfile, контексты и зависимости Jakarta Компоненты ввода отслеживаются, если присутствует аннотация «org.eclipse.microprofile.opentracing.Traced» выполняется либо на уровне типа, либо на уровне метода. Трассировку можно отключить, установив для аннотации значение «false». Аналогично, пользовательское имя операции можно задать, указав параметр «OperationName» для этой аннотации. Семантика определяется спецификацией OpenTracing для MicroProfile.

Обратите внимание, что для активации поддержки OpenTracing в MicroProfile требуется поддержка контекстов Jakarta и внедрения зависимостей для развертывания. Простое приложение Jakarta RESTful Web Services без контекстов Jakarta и поддержки внедрения зависимостей не будет отслеживаться.

Управляемые компоненты с несколькими аспектами, такие как компоненты Jakarta Enterprise, также можно отслеживать, помечая их с помощью @Traced, но с ограничениями. Например, асинхронные вызовы приведут к созданию новой трассировки вместо того, чтобы иметь диапазон для присоединения к существующей трассировке.

### 7.18.4 Ссылка на компонент

Реализация MicroProfile OpenTracing предусмотрена проектом SmallRye OpenTracing.

- `MicroProfile OpenTracing`
- `SmallRye OpenTracing`

## 7.19 Конфигурация подсистемы обеспечения отказоустойчивости MicroProfile

Поддержка отказоустойчивости MicroProfile обеспечивается также подсистемой «microprofile-fault-tolerancesmallrye».

### 7.19.1 Требуемое расширение

Это расширение по умолчанию не включено в стандартные конфигурации дистрибутива WildBoss Pro.

Можно добавить расширение в конфигурацию без него, либо добавив <extension module="org.WildBoss Pro.extension.microprofile.fault-tolerance-smallrye"/> в формате XML или с помощью следующей операции CLI:

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.microprofile.faulttolerance-smallrye:add()
{"outcome" => "success"}

[standalone@localhost:9990 /] /subsystem=microprofile-fault-tolerance-
smallrye:add()
{
   "outcome" => "success",
   "response-headers" => {
      "operation-requires-reload" => true,
      "process-state" => "reload-required"
   }
}
[standalone@localhost:9990 /] reload
```

Подсистема не имеет каких-либо настраиваемых элементов.

### 7.19.2 Спецификация

Подсистема обеспечения отказоустойчивости MicroProfile WildBoss Pro реализует версию MicroProfile 3.0.

В спецификации MicroProfile предусмотрены следующие привязки перехватчика:

– @Timeout;
– @Retry;
– @Fallback;
– @CircuitBreaker;
– @Bulkhead;
– @Asynchronous.

Для удобства использования, пожалуйста, ознакомьтесь со спецификацией MicroProfile Fault Tolerance 3.0.

### 7.19.3 Конфигурация

Помимо свойств конфигурации, определенных спецификацией, реализация SmallRye предоставляет следующие свойства конфигурации:

Таблица 9 - Свойства конфигурации малой отказоустойчивости

| Свойство | Значение по умолчанию | Описание |
|---|---|---|
| io.smallrye.faulttolerance.mainThreadPoolSize | 100 | Максимальное количество потоков в пуле потоков |
| io.smallrye.faulttolerance.mainThreadPoolQueueSize | -1 (unbounded) | Размер очереди, которую должен использовать пул потоков |

### 7.19.4 Ссылка на компонент

Реализация отказоустойчивости MicroProfile обеспечивается проектом SmallRye Fault Tolerance project.

```
• MicroProfile Fault Tolerance
• SmallRye Fault Tolerance
```

## 7.20 Конфигурация подсистемы операторы реактивных потоков MicroProfile

Поддержка операторов MicroProfile реактивных потоков предоставляется в качестве функции технического просмотра подсистемой «microprofile-reactive-streams-operators-smallrye».

### 7.20.1 Требуемое расширение

Это расширение не входит в стандартные конфигурации, включенные в дистрибутив WildBoss Pro.

Можно добавить расширение в конфигурацию, либо добавив элемент <extension module="org.WildBoss Pro.extension.microprofile.reactive-streams-operators-smallrye"/> в «xml», либо используя следующую операцию CLI:

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.microprofile.reactivestreams-operators-smallrye:add
{"outcome" => "success"}

[standalone@localhost:9990 /] /subsystem=microprofile-reactive-streams-
operatorssmallrye:
add
{
   "outcome" => "success",
   "response-headers" => {
      "operation-requires-reload" => true,
      "process-state" => "reload-required"
   }
}
```

Если вы создадите свой собственный сервер и включите уровень «microprofile-reactive-streams-operators», вы получите необходимые модули, а расширение и подсистема будут добавлены в вашу конфигурацию.

### 7.20.2 Спецификация

Подсистема операторов MicroProfile реактивных потоков WildBoss Pro реализует MicroProfile реактивные потоки Streams Operators 2.0, который добавляет поддержку асинхронной потоковой передачи данных. Он, по сути, копирует интерфейсы и их реализации, которые были доступны в классе «java.util.concurrent.Flow» появился в Java 9. Таким образом, операторы реактивных потоков с MicroProfile можно считать промежуточным этапом до тех пор, пока Java 9 и более поздние версии не станут повсеместными.

### 7.20.3 Конфигурация

Подсистема «microprofile-reactive-streams-operators-smallrye» не содержит настраиваемых атрибутов или ресурсов. Ее наличие делает интерфейсы из MicroProfile

Reactive Streams Operators доступными для развертывания и обеспечивает реализацию. Кроме того, это делает экземпляр класса «ReactiveStreamsEngine» доступным для внедрения.

Активация: если подсистема присутствует, функциональность операторов реактивных потоков MicroProfile будет доступна для всех развертываний на сервере.

### 7.20.4 Ссылка на компонент

Внедрение MicroProfile операторов реактивных потоков обеспечивается проектом SmallRye Mutiny.

- MicroProfile Reactive Streams Operators
- SmallRye Mutiny

## 7.21 Конфигурация подсистемы реактивного обмена сообщениями с MicroProfile

```
:smallrye-reactive-messaging-version: 3.6
:smallrye-reactive-messaging-tag: {smallrye-reactive-messaging-
version}.0
:eclipse-mp-reactive-messaging-api-version: 2.0
```

Поддержка MicroProfile реактивных сообщений предоставляется в качестве функции технического просмотра подсистемой «microprofile-reactive-messaging-smallrye».

### 7.21.1 Требуемое расширение

Это расширение не входит в стандартные конфигурации, включенные в дистрибутив WildBoss Pro.

Можно добавить расширение в конфигурацию, либо добавив элемент <extension module="org.WildBoss Pro.extension.microprofile.reactive-messaging-smallrye"/> в «xml», либо используя следующую операцию CLI:

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.microprofile.reactivemessaging-smallrye:add
{"outcome" => "success"}
[standalone@localhost:9990 /] /subsystem=microprofile-reactive-
messaging-smallrye:add
{
   "outcome" => "success",
   "response-headers" => {
      "operation-requires-reload" => true,
      "process-state" => "reload-required"
   }
}
```

Чтобы использовать эту подсистему, также должно быть включено расширение и подсистема операторов реактивных потоков MicroProfile.

Если вы создадите свой собственный сервер и включите уровень «Galleon» с «microprofile-reactive-messaging», вы получите необходимые модули, а расширение и подсистема будут добавлены в вашу конфигурацию.

Если вы предоставляете уровень Galleon для «microprofile-reactive-messaging-kafka», он включает в себя модули для включения функциональности «Kafka connector». Уровень «microprofile-reactive-messaging-kafka» включает в себя уровень «microprofile-reactive-

messaging», который обеспечивает основную функциональность MicroProfile функции реактивного обмена сообщениями.

### 7.21.2 Спецификация

Подсистема MicroProfile Reactive Messaging WildBoss Pro реализует MicroProfile Reactive Messaging «{eclipse-mp-reactive-messaging-api-версия}», которая добавляет поддержку асинхронных сообщений на основе операторов MicroProfile Reactive Streams Operators.

### 7.21.3 Конфигурация

Подсистема «microprofile-reactive-messaging-smallrye» не содержит настраиваемых атрибутов или ресурсов. Для основной функциональности MicroProfile Reactive Messaging настройка отсутствует. Для настройки соединителей с внешними брокерами используется конфигурация MicroProfile Config.

*Активация* - подсистема проверит все развертывания, чтобы найти классы, содержащие методы с аннотациями «org.eclipse.microprofile.reactive.messaging.Incoming» или «org.eclipse.microprofile.reactive.messaging.Outgoing» аннотации. Если эти примечания будут найдены, для развертывания будет включена функция реактивного обмена сообщениями.

*Модель программирования и ограничения* - для получения более подробных примеров необходимо смотреть спецификацию, в этом разделе просто предпринята попытка обобщить основные моменты.

В версии 1.0 спецификации реактивного обмена сообщениями MicroProfile были добавлены аннотации «@Incoming» и «@Outgoing». Они предназначены для использования в компоненте CDI «@ApplicationScoped» (или «@Dependent»):

```
@ApplicationScoped
public class MyBean {
   @Outgoing("in-memory")
   public String generate() {
        return ...; // Do some generation of values
   }
   @Incoming("in-memory")
   public void consume(String value) {
        System.out.println(value);
   }
}
```

Значения, сгенерированные методом «generate()», будут получены методом «consume()». В этой базовой настройке, когда имена каналов совпадают, потоки обрабатываются в памяти. Позже можно увидеть, как с ними обращаться в Kafka.

В приведенном выше примере, по сути, генерируются значения и используют их без участия пользователя. В MicroProfile Reactive Messaging 2.0 добавлена аннотация «@Channe»l, которая может использоваться для добавления издателя для получения значений, отправляемых в потоках, и отправителя, который может использоваться для отправки значений в поток. Это упрощает отправку/получение значений из путей кода, возникающих в результате взаимодействия с пользователем:

```
@ApplicationScoped
public class MyBean {
   @Inject
   @Channel("in-memory")
   Emitter<String> emitter;
```

```
    @Inject
    @Channel("in-memory")
    Publisher<String> publisher;
    void send(String value) {
        emitter.send(value);
    }
}
```

В приведенном выше примере мы теперь можем легко отправлять данные в потоки реактивных сообщений, вызывая Emitter.send(). Аналогичным образом мы можем подписаться на публикатора и получать данные. Однако у получения все еще есть несколько недостатков:

– приведенный выше пример не будет работать в готовом виде. При попытке отправки от отправителя вы получите сообщение об ошибке, что нет подписчиков (что, в свою очередь, может привести к переполнению). Это можно обойти, создав подписку на Publisher;

– в настоящее время у введенного издателя может быть только одна подписка.

Вышеуказанные пункты означают, что этот издатель не может использоваться напрямую в качестве асинхронного возвращаемого значения, например, для конечной точки веб-сервисов RESTful в Jakarta. Поскольку запрос Jakarta RESTful Webservices — это то, что создаст подписку, такой вызов должен был бы произойти перед вызовом функции Emitter.send().

Если заменить отправителя на метод generate() из исходного метода, то данный пример будет работать. Однако, если вернуть издателя более чем на один запрос Jakarta RESTful Webservices, то получим несколько подписок, которые не будут получать каждое отдельное значение.

Пользовательские приложения, предназначенные для возврата опубликованных значений пользователям, например, через Jakarta RESTful Веб-сервисы должны будут выполнять свои собственные подписки и буферизацию данных. Необходимо соблюдать осторожность, чтобы не допустить неконтролируемого увеличения объема кэша, что может привести к ошибкам «OutOfMemoryErrors».

*Соединители* - модуль MicroProfile Reactive Messaging разработан таким образом, чтобы быть достаточно гибким для интеграции с широким спектром внешних систем обмена сообщениями. Эта функциональность предоставляется через «соединители».

На данный момент единственным входящим в комплект разъемом является разъем Kafka.

Соединители настраиваются с помощью MicroProfile Config. Ключи свойств для методов имеют некоторые префиксы, предписанные спецификацией реактивного обмена сообщениями MicroProfile, в которой они перечислены следующим образом:

– `mp.messaging.incoming.[channel-name].[attribute]=[value];`

– `mp.messaging.outgoing.[channel-name].[attribute]=[value];`

– `mp.messaging.connector.[connector-name].[attribute]=[value].`

По сути, имя канала — это «@Incoming.value()» или «@Outcoming.value()».

Если есть следующая пара методов:

```
@Outgoing("to")
public int send() {
int i = // Randomly generated...
}
@Incoming("from")
public void receive(int i) {
// Process payload
}
```

Тогда префиксы свойств, предписанные спецификациями MicroProfile Reactive Messaging, будут следующими:

– `mp.messaging.incoming.from.` - это позволило бы выбрать свойство в качестве конфигурации функции receive() метод;

– `mp.messaging.outgoing.to.` - это позволило бы выбрать свойство в качестве конфигурации функции send() метод.

Обратите внимание, что, хотя эти префиксы понятны подсистеме, полный набор зависит от того, какой соединитель вы хотите настроить. Разные соединители понимают разные свойства.

*Соединитель Kafka* - пример минимального файла «microprofile-config.properties» для Kafka для примера приложения, показанного ранее:

```
kafka.bootstrap.servers=kafka:9092
mp.messaging.outgoing.to.connector=smallrye-kafka
mp.messaging.outgoing.to.topic=my-topic
mp.messaging.outgoing.to.value.serializer=org.apache.kafka.common.seria
lization.Intege
rSerializer
mp.messaging.incoming.from.connector=smallrye-kafka
mp.messaging.incoming.from.topic=my-topic
mp.messaging.incoming.from.value.deserializer=org.apache.kafka.common.s
erialization.In
tegerDeserializer
```

Далее мы кратко обсудим каждую из этих записей. Помните, что канал «to» используется в методе send() , а канал «from» - в методе receive().

`kafka.bootstrap.servers=kafka:9092` устанавливает URL-адрес посредника Kafka, к которому будет подключаться все приложение. Это также можно было бы сделать только для канала «to», установив вместо этого `mp.messaging.outgoing.to.bootstrap.servers=kafka:9092`.

`mp.messaging.outgoing.to.connector=smallrye-kafka` говорит о том, что мы хотим использовать Kafka для поддержки канала «to». Обратите внимание, что значение «smallrye-kafka» относится к реактивным сообщениям SmallRye и будет понято только в том случае, если включен коннектор Kafka.

`mp.messaging.outgoing.to.topic=my-topic` говорит, что мы отправим данные в тему «Kafka» под названием «mytopic».

`mp.messaging.outgoing.to.value.serializer=org.apache.kafka.com mon.serialization.IntegerSerializer` указывает соединителю использовать IntegerSerializer для сериализации значений, выводимых методом send() при записи в раздел. Kafka предоставляет сериализаторы для стандартных типов Java. Вы можете реализовать свой собственный сериализатор, написав класс, реализующий «org.apache.kafka.common.serialization.Serializer» и включив его в развертывание.

`mp.messaging.incoming.from.connector=smallrye-kafka` говорит о том, что возможно использовать Kafka для поддержки канала «from». Как указано выше, значение «smallrye-kafka» относится к реактивным сообщениям SmallRye.

`mp.messaging.incoming.from.topic=my-topic` говорит, что мы будем считывать данные из темы «Kafka» под названием «my-topic».

`mp.messaging.incoming.from.value.deserializer=org.apache.kafka .common.serialization.IntegerDeserializer` указывает соединителю использовать IntegerDeserializer для десериализации значений из раздела перед вызовом метода receive(). Можно реализовать собственный десериализатор, написав класс,

реализующий «org.apache.kafka.common.serialization.Deserializer» и включение его в процесс развертывания.

В дополнение к вышесказанному, Apache Kafka и Kafka connector от SmallRye Reactive Messaging поддерживают гораздо больше свойств. Их можно найти в документации по SmallRye Reactive Messaging Kafka connector, а также в документации по Apache Kafka для производителей и потребителей.

Описанные выше префиксы удаляются перед передачей свойства в Kafka. То же самое происходит и с другими свойствами конфигурации. Смотрите документацию по Kafka для получения более подробной информации о том, как настроить пользователей и производителей Kafka.

*Подключение к защищенному Kafka* - при подключении к экземпляру Kafka, защищенному с помощью SSL и SASL, следующий пример «microprofileconfig.properties» поможет вам начать работу. Добавлено несколько новых свойств. Мы показываем их на уровне соединителя, но они с таким же успехом могут быть определены на уровне канала (т.е. с помощью «mp.messaging.outgoing.to-kafka.» и «mp.messaging.incoming.from-kafka.» префиксы из предыдущих примеров, а не префикс «mp.messaging.connector.smallrye-kafka» для всего соединителя).

```
mp.messaging.connector.smallrye-kafka.bootstrap.servers=localhost:9092
mp.messaging.connector.smallrye-kafka.sasl.mechanism=PLAIN
mp.messaging.connector.smallrye-kafka.security.protocol=SASL_SSL
mp.messaging.connector.smallryekafka.
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModul
e
required \
username="${USER}" \
password="${PASSWORD}";
mp.messaging.connector.smallrye-kafka.WildBoss
Pro.elytron.ssl.context=test
# Channel configuration would follow here, but is left out for brevity
```

Каждая из этих строк имеет следующее значение:

– `mp.messaging.connector.smallrye-kafka.bootstrap.servers= localhost:9092` - указывает серверы Kafka , к которым нужно подключаться. Это то же самое, что и в предыдущих примерах;

– `mp.messaging.connector.smallrye-kafka.sasl.mechanism=PLAIN` - определяет используемый механизм SASL. Другие варианты смотрите в разделе «sasl.mechanism» в документации Kafka;

– `mp.messaging.connector.smallrye-kafka.security.protocol` - определяет используемый механизм протокола. Другие варианты смотрите в разделе «security.protocol» в документации по Kafka. В данном случае мы используем SASL_SSL, что означает, что связь осуществляется по протоколу SSL и что SASL используется для аутентификации;

– `mp.messaging.connector.smallrye-kafka.sasl.jaas.config=…` - указывает, как мы будем проходить аутентификацию с помощью Kafka. Чтобы не вводить жестко учетные данные в наш файл «microprofile-config.properties», мы используем функцию замены свойств в MicroProfile Config. В этом случае, если вы определили переменные среды USER и PASSWORD, они будут переданы как часть конфигурации;

– `mp.messaging.connector.smallrye-kafka.WildBoss Pro.elytron.ssl. context=test` - это не требуется, если Kafka защищен сертификатом, подписанным CA. Если используются самозаверяющие сертификаты, нужно будет указать хранилище доверия в подсистеме Elytron и создать SSLContext, ссылающийся на него. Значение этого свойства используется для поиска SSLContext в подсистеме Elytron в

разделе «/subsystem=elytron/client-ssl-context=*» в модели управления WildBoss Pro. В этом случае значением свойства является «test», поэтому мы ищем SSLContext, определенный с помощью «/subsystem=elytron/client-sslcontext= test», и используем его для настройки хранилища доверия для подключения к Kafka.

*Пользовательский API Kafka* - чтобы иметь возможность получать больше информации о сообщениях, полученных от Kafka, и влиять на то, как Kafka обрабатывает сообщения, для Kafka существует пользовательский API. Этот API находится в пакете «io/smallrye/reactive/messaging/kafka/api».

API состоит из следующих классов:

– IncomingKafkaRecordMetadata - эти метаданные содержат такую информацию, как:

– ключ к записи Kafka, представленный сообщением;

– тема и раздел Kafka, используемые для сообщения, и смещение внутри них;

– временная метка сообщения и тип временной метки;

– заголовки сообщений — это фрагменты информации, которые приложение может прикреплять на стороне-отправителе и получать на стороне-потребителе. Они сохраняются и пересылаются Kafka, но не имеют значения для самой Kafka;

– OutgoingKafkaRecordMetadata - это создается с помощью метода «builder», возвращаемого с помощью метода builder() и позволяет вам указать/переопределить, как Kafka будет обрабатывать сообщения. Аналогично случаю IncomingKafkaRecordMetadata, можно установить:

– ключ. Затем Kafka будет рассматривать эту запись как ключ к сообщению;

– тема, как уже было показано, обычно используется конфигурацию «microprofile-config.propertie», чтобы указать тему, которая будет использоваться для канала, поддерживаемого Kafka. Однако в некоторых случаях коду, отправляющему сообщение, может потребоваться сделать некоторый выбор (например, в зависимости от значений, содержащихся в данных) относительно того, в какую тему отправлять сообщение. Если указать это здесь, Kafka будет использовать эту тему;

– раздел. Как правило, лучше всего позволить разделителю Kafka выбрать раздел, но в случаях, когда важно иметь возможность указать его, это можно сделать;

– временная метка, если необходимо, чтобы она автоматически генерировалась Kafka;

– заголовки - вы можете прикрепить заголовки для пользователя, как указано для IncomingKafkaRecordMetadata;

– KafkaMetadataUtil содержит служебные методы для записи исходящих «CAFKARECORDMETADATA» в сообщение и для чтения входящих Cafkarecordmetadata из сообщения. Обратите внимание, что если вы пишете Исходящие данные Cafkarecordmetadata для сообщения, отправленного по каналу, который не обрабатывается Kafka, будут проигнорированы, и, если вы попытаетесь прочитать входящие данные Cafkarecordmetadata из сообщения, поступающего по каналу, который не обрабатывается Kafka, значение будет равно «null».

В следующем примере показано, как записать и прочитать ключ из сообщения:

```
@Inject
@Channel("from-user")
Emitter<Integer> emitter;
@Incoming("from-user")
@Outgoing("to-kafka")
public Message<Integer> send(Message<Integer> msg) {
     // Set the key in the metadata
     OutgoingKafkaRecordMetadata<String> md =
     OutgoingKafkaRecordMetadata.<String>builder()
     .withKey("KEY-" + i)
     .build();
```

```
        // Note that Message is immutable so the copy returned by this
        method
        // call is not the same as the parameter to the method
        return KafkaMetadataUtil.writeOutgoingKafkaMetadata(msg, md);
    }
    @Incoming("from-kafka")
    public CompletionStage<Void> receive(Message<Integer> msg) {
        IncomingKafkaRecordMetadata<String, Integer> metadata =
            KafkaMetadataUtil.readIncomingKafkaMetadata(msg).get();
        // We can now read the Kafka record key
        String key = metadata.getKey();
        // When using the Message wrapper around the payload we need to
        explicitly ack
        // them
        return msg.ack();
    }
```

Чтобы настроить отображение Kafka, нам нужен «microprofile-config.properties»

```
kafka.bootstrap.servers=kafka:9092
mp.messaging.outgoing.to-kafka.connector=smallrye-kafka
mp.messaging.outgoing.to-kafka.topic=some-topic
mp.messaging.outgoing.tokafka.
value.serializer=org.apache.kafka.common.serialization.IntegerSerialize
r
mp.messaging.outgoing.tokafka.
key.serializer=org.apache.kafka.common.serialization.StringSerializer
mp.messaging.incoming.from-kafka.connector=smallrye-kafka
mp.messaging.incoming.from-kafka.topic=some-topic
mp.messaging.incoming.fromkafka.
value.deserializer=org.apache.kafka.common.serialization.IntegerDeseria
lizer
mp.messaging.incoming.from-
kafka.key.deserializer=org.apache.kafka.common.serialization.StringDese
rializer
```

Эта конфигурация очень похожа на предыдущую конфигурацию, которую мы видели, но обратите внимание, что нам нужно указать «key.serializer» для исходящего канала и «key.deserializer» для входящего канала. Как и прежде, это реализации «org.apache.kafka.common.serialization.Serializer» и «org.apache.kafka.common.serialization.Deserializer» соответственно. Kafka предоставляет реализации для базовых типов, и вы можете написать свои собственные и включить их в развертывание.

Заметка об «org.apache.kafka» классы.

Хотя мы и предоставляем клиентский jar-файл Kafka в наших спецификациях, его использование ограничено:

– классы/интерфейсы, доступные через пользовательский API Kafka, например:
  – «org.apache.kafka.common.header.Header» и «org.apache.kafka.common.header.Headers» и реализации тех, которые считаются общедоступными API в соответствии с документацией Apache Kafka;
  – «org.apache.kafka.clients.consumer.ConsumerRecord»;
  – «org.apache.kafka.common.record.TimestampType»;
– классы/интерфейсы, необходимые для сериализации и десериализации:
  – org.apache.kafka.common.serialization.Deserializer;
  – org.apache.kafka.common.serialization.Serializer;

– реализации «org.apache.kafka.common.serialization.Deserializer» и «org.apache.kafka.common.serialization.Serializer» в пакет «org.apache.kafka.common.serialization».

### 7.21.4 Ссылка на компонент

В MicroProfile Reactive Messaging обеспечивается SmallRye Reactive Messaging Проект обмена сообщениями.

```
• MicroProfile Reactive Messaging
• SmallRye Reactive Messaging
```

## 7.22 Настройка веб-служб

Компоненты JBossWS предоставляются серверу приложений через подсистему веб-сервисов. Компоненты JBossWS обрабатывают конечные точки WS. Подсистема поддерживает настройку опубликованных адресов конечных точек и цепочки обработчиков конечных точек. Подсистема веб-сервисов по умолчанию предоставляется в домене сервера и автономных файлах конфигурации.

### 7.22.1 Структура подсистемы веб-сервисов

*Опубликованный адрес конечной точки* - JBossWS поддерживает перезапись элемента <soap:address> для конечных точек, опубликованных в контрактах WSDL. Эта функция полезна для управления адресом сервера, который сообщается клиентам для каждой конечной точки.

Следующие элементы (таблица 10) доступны и могут быть изменены (все они необязательны):

Таблица 10 – Элементы доступные для изменения

| Имя | Тип | Описание |
|---|---|---|
| modify-wsdl-address | логический | Это логическое значение включает и отключает функцию перезаписи адреса. Если для параметра «modifywsdl-address» установлено значение «true» и содержимое <soap:address> является допустимым URL-адресом, JBossWS перепишет URL-адрес, используя значения «wsdl-host» и «wsdlport» или «wsdl-secure-port». Если для параметра «modify-wsdl-address» задано значение «false», а содержимое <soap:address> является допустимым URL-адресом, JBossWS не будет переписывать URL-адрес. Будет использоваться URL-адрес <soap:address> .Если содержимое <soap:address> не является допустимым URL, JBossWS перепишет его независимо от значения параметра «modify-wsdl-address». Если для параметра «modifywsdl- address» задано значение «true», а для параметра «wsdl-host» не определено или явно задано значение «jbossws.undefined.host» , используется |

| Имя | Тип | Описание |
|---|---|---|
| | | содержимое URL-адреса <soap:address> . JBossWS использует хост отправителя запроса при перезаписи <soap:address>, когда «modify-wsdl-address» не определен, JBossWS использует значение по умолчанию «true» |
| wsdl-host | строка | Имя хоста / IP-адрес, которые будут использоваться для перезаписи <soap:address>.Если для параметра «wsdl-host» задано значение «jbossws.undefined.host», JBossWS использует хост отправителя запроса при перезаписи <soap:address>, когда wsdl-host не определен, JBossWS использует значение по умолчанию «jbossws.undefined.host» |
| wsdl-port | целочисленный тип данных | Задайте это свойство для явного определения HTTP-порта, который будет использоваться для перезаписи SOAP -адреса.В противном случае HTTP -порт будет идентифицирован путем запроса списка установленных HTTP-соединителей |
| wsdl-secure-port | целочисленный тип данных | Задайте это свойство для явного определения HTTPS-порта, который будет использоваться для перезаписи SOAP -адреса. В противном случае HTTPS -порт будет идентифицирован путем запроса списка установленных HTTPS-коннекторов |
| wsdl-uri-scheme | строка | Это свойство явно задает схему URI, используемую для перезаписи <soap:address> . Допустимыми значениями являются «http» и «https». Эта конфигурация переопределяет схему, вычисленную путем обработки конечной точки (даже если указана гарантия передачи). Указанные значения для «wsdl-port» и «wsdl-secure-port» (или их значения по умолчанию) используются в зависимости от указанной схемы |
| wsdl-path-rewrite-rule | строка | Эта строка определяет команду подстановки SED (например, «s/regexp/replacement/g»), которую JBossWS выполняет для компонента «path» каждого <soap:address> URL-адреса, опубликованного с сервера. Если «wsdlpath-rewrite-rule» не определено, JBossWS сохраняет исходный компонент «path» для каждого URL-адреса <soap:address>.Если для параметра «modify-wsdl-address» задано значение «false», этот элемент игнорируется |

Предопределенные конфигурации конечных точек - JBossWS позволяет предварительно определять дополнительные данные конфигурации установки и связывать их с реализацией конечной точки. Предварительно определенные конфигурации конечной точки можно использовать для клиента веб-служб Jakarta XML и для настройки конечной точки веб-служб Jakarta XML. Конфигурации конечной точки могут включать в себя Jakarta Обработчики веб-служб XML и объявления свойств ключа/значения. Эта функция

предоставляет удобный способ добавления обработчиков к конечным точкам WS и задания свойств ключа/значения, которые управляют Внутренними компонентами JBossWS и Apache CXF (см. раздел «Конфигурация Apache CXF»).

Подсистема веб-сервисов предоставляет схему для поддержки определения именованных наборов данных конфигурации конечной точки. Аннотация, org.jboss.ws.api.annotation.EndpointConfig предоставляется для сопоставления именованной конфигурации с реализацией конечной точки.

Количество конфигураций конечных точек, которые могут быть определены в подсистеме веб-сервисов, не ограничено. Каждая конфигурация конечной точки должна иметь уникальное имя в подсистеме веб-сервисов. Конфигурации конечных точек, определенные в подсистеме веб-сервисов, доступны для ссылки по имени в аннотации к любой конечной точке в развернутом приложении.

WildBoss Pro поставляется с двумя предопределенными конфигурациями конечных точек. Конфигурация по умолчанию - Standard-Endpoint-Config. Recording-Endpoint-Config является примером пользовательской конфигурации конечной точки и включает в себя обработчик записи.

```
[standalone@localhost:9999 /] /subsystem=webservices:read-resource
{
  "outcome" => "success",
  "result" => {
    "endpoint" => {},
    "modify-wsdl-address" => true,
    "wsdl-host" => expression "${jboss.bind.address:127.0.0.1}",
    "endpoint-config" => {
       "Standard-Endpoint-Config" => undefined,
       "Recording-Endpoint-Config" => undefined
    }
  }
}
```

**Внимание:** Standard-Endpoint-Config — это специальная конфигурация конечной точки. Она используется для любой конечной точки, для которой явно не назначена конфигурация конечной точки.

*Конфигурации конечных точек* определяются с помощью элемента «endpoint-config». Каждая конфигурация конечной точки может включать свойства и обработчики, установленные для конечных точек, связанных с конфигурацией.

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-
config=Recording-Endpoint-Config:read-resource
{
  "outcome" => "success",
  "result" => {
     "post-handler-chain" => undefined,
     "property" => undefined,
     "pre-handler-chain" => {"recording-handlers" => undefined}
  }
}
```

Новая конфигурация конечной точки может быть добавлена следующим образом:

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-
config=My-Endpoint-Config:add
{
  "outcome" => "success",
  "response-headers" => {
```

```
      "operation-requires-restart" => true,
      "process-state" => "restart-required"
    }
}
```

*Цепи погрузчика* - каждая конфигурация конечной точки может быть связана с нулем или более цепочками обработчиков «ДО» и «ПОСЛЕ» обработки. Каждая цепочка обработчиков может включать обработчики JAXWS. Для исходящих сообщений цепочки предварительных обработчиков выполняются перед любым обработчиком, который подключен к конечной точке с использованием стандартных средств, таких как аннотация «@HandlerChain», а цепочки обработчиков POST выполняются после выполнения этих объектов. Для входящих сообщений цепочки обработчиков POST выполняются перед любым обработчиком, который подключен к конечной точке стандартными средствами, а цепочки предварительных обработчиков выполняются после выполнения этих объектов.

```
* Server inbound messages
Client --> ... --> POST HANDLER --> ENDPOINT HANDLERS --> PRE
HANDLERS --> Endpoint
* Server outbound messages
Endpoint --> PRE HANDLER --> ENDPOINT HANDLERS --> POST HANDLERS -->
... --> Client
```

Атрибут привязки к протоколу должен использоваться для установки протоколов, для которых будет запущена цепочка.

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-
config=Recording-Endpoint-Config/pre-handler-chain=recording-
handlers:read-resource
{
  "outcome" => "success",
  "result" => {
  "protocol-bindings" => "##SOAP11_HTTP ##SOAP11_HTTP_MTOM
  ##SOAP12_HTTP
  ##SOAP12_HTTP_MTOM",
  "handler" => {"RecordingHandler" => undefined}
  },
  "response-headers" => {"process-state" => "restart-required"}
}
```

Новая цепочка обработчиков может быть добавлена следующим образом:

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-
config=My-Endpoint-Config/post-handler-chain=my-
handlers:add(protocol-bindings="##SOAP11_HTTP")
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-restart" => true,
    "process-state" => "restart-required"
  }
}
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-
config=My-Endpoint-Config/post-handler-chain=my-handlers:read-
resource
{
  "outcome" => "success",
```

```
    "result" => {
        "handler" => undefined,
        "protocol-bindings" => "##SOAP11_HTTP"
    },
    "response-headers" => {"process-state" => "restart-required"}
}
```

***Обработчики.***

Обработчик JAXWS может быть добавлен в цепочки обработчиков:

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-
config=Recording-Endpoint-Config/pre-handler-chain=recording-
handlers/handler=RecordingHandler:readresource
{
    "outcome" => "success",
    "result" => {"class" =>
    "org.jboss.ws.common.invocation.RecordingServerHandler"},
    "response-headers" => {"process-state" => "restart-required"}
}
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-
config=My-Endpoint-Config/post-handler-chain=my-
handlers/handler=foohandler:add(class="org.jboss.ws.common.invocation
.RecordingServerHandler")
{
    "outcome" => "success",
    "response-headers" => {
    "operation-requires-restart" => true,
    "process-state" => "restart-required"
    }
}
```

Загрузка класса обработчика конфигурации конечной точки.

**Внимание:** атрибут «class» используется для указания полного имени класса обработчика. Во время развертывания создается экземпляр класса для каждого ссылающегося развертывания. Для успешного создания класса загрузчик классов развертывания должен иметь возможность загружать класс обработчика.

### 7.22.2 Информация о времени выполнения

Каждая конечная точка веб-службы доступна через развертывание, которое обеспечивает реализацию конечной точки. Каждая конечная точка может быть запрошена как ресурс развертывания. Для получения дополнительной информации обратитесь к главе «Развертывание приложений». Каждая конечная точка веб-службы определяет веб-контекст и URL-адрес WSDL:

```
[standalone@localhost:9999 /]
/deployment="*"/subsystem=webservices/endpoint="*":readresource
{
    "outcome" => "success",
    "result" => [{
        "address" => [
            ("deployment" => "jaxws-samples-handlerchain.war"),
            ("subsystem" => "webservices"),
            ("endpoint" => "jaxws-samples-handlerchain:TestService")
        ],
        "outcome" => "success",
```

```
      "result" => {
        "class" =>
        "org.jboss.test.ws.jaxws.samples.handlerchain.EndpointImpl",
        "context" => "jaxws-samples-handlerchain",
        "name" => "TestService",
        "type" => "JAXWS_JSE",
        "wsdl-url" => "http://localhost:8080/jaxws-samples-
        handlerchain?wsdl"
      }
    }]
  }
```

### 7.22.3 Ссылка на компонент

Подсистема веб-сервисов предоставляется проектом JBossWS. Для получения подробного описания доступных свойств конфигурации, пожалуйста, обратитесь к документации проекта.

```
• JBossWS homepage: http://www.jboss.org/jbossws
• Project Documentation: https://docs.jboss.org/author/display/JBWS
```

## 7.23 Адаптеры ресурсов

Адаптеры ресурсов настраиваются через подсистему адаптеров ресурсов. Объявление нового адаптера ресурсов состоит из двух отдельных шагов: вам потребуется развернуть архив «*.rar» и определить запись адаптера ресурсов в подсистеме.

### 7.23.1 Определения адаптеров ресурсов

Сам адаптер ресурсов определяется в подсистеме «resource-adapters» (ресурсы-адаптеры):

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.0">
   <resource-adapters>
      <resource-adapter>
         <archive>eis.rar</archive>
         <!-- Resource adapter level config-property -->
         <config-property name="Server">localhost</config-property>
         <config-property name="Port">19000</config-property>
         <transaction-support>XATransaction</transaction-support>
         <connection-definitions>
            <connection-definition class-name=
            "com.acme.eis.ra.EISManagedConnectionFactory"
                        jndi-name="java:/eis/AcmeConnectionFactory"
                        pool-name="AcmeConnectionFactory">
               <!-- Managed connection factory level config-property -->
               <config-property name="Name">Acme Inc</config-property>
```

```
                <pool>
                    <min-pool-size>10</min-pool-size>
                    <max-pool-size>100</max-pool-size>
                </pool>
                <security>
                    <application/>
                </security>
                </connection-definition>
            </connection-definitions>
            <admin-objects>
                <admin-object class-name="com.acme.eis.ra.EISAdminObjectImpl"
                              jndi-name="java:/eis/AcmeAdminObject">
                    <config-property name="Threshold">10</config-property>
                </admin-object>
            </admin-objects>
        </resource-adapter>
    </resource-adapters>
</subsystem>
```

Обратите внимание, что поддерживаются только привязки JNDI под «java:/» или «java:jboss/». (см. «standalone/configuration/standalone.xml»)

### 7.23.2 Автоматическая активация архивов адаптеров ресурсов

Архив адаптера ресурсов можно автоматически активировать с помощью конфигурации, включив в архив символ «META-INF/ironjacamar.xml».

Схему можно найти по адресу «http://docs.jboss.org/ironjacamar/schema/ironjacamar_1_0.xsd»

### 7.23.3 Ссылка на компонент

Подсистема адаптера ресурсов предоставляется проектом IronJacamar. Для получения подробного описания доступных свойств конфигурации, пожалуйста, обратитесь к документации проекта.

– домашняя страница IronJacamar: http://www.jboss.org/ironjacamar;

– проектная документация: http://www.jboss.org/ironjacamar/docs;

– описание схемы: http://docs.jboss.org/ironjacamar/userguide/1.0/en-US/html/deployment.html#deployingra_descriptor.

## 7.24 Конфигурация пакетной подсистемы в Jakarta

Пакетная подсистема используется для настройки среды для запуска пакетных приложений. WildBoss Pro использует JBeret для пакетной реализации. Более подробную информацию о JBeret можно найти в руководстве пользователя. Путь к ресурсу в нотации CLI для подсистемы выглядит следующим образом: «subsystem=batch-jberet».

### 7.24.1 Конфигурация подсистемы по умолчанию

Более подробную информацию о параметрах конфигурации подсистемы смотрите в справочнике по модели WildBoss Pro.

### 7.24.2 Безопасность

В подсистему «batch-jberet» был добавлен новый атрибут «security-domain», позволяющий выполнять пакетные задания в этом домене безопасности. Задания, которые были остановлены в рамках операции приостановки, будут перезапущены при возобновлении работы с первоначальным пользователем, запустившим задание.

Был добавлен «org.WildBoss Pro.extension.batch.jberet.deployment». Добавлена функция «BatchPermission», позволяющая ограничить безопасность различных пакетных функций. С этим разрешением можно управлять следующими функциями:

– start;
– stop;
– restart;
– abandon;
– read.

Функция чтения позволяет пользователям использовать методы получения данных из «javax.batch.operations.JobOperator» или прочитать «batch-jberet» ресурс развертывания, например «/deployment=my.war/subsystem=batch-jberet:read-resource».

### 7.24.3 Хранилище заданий

Пакетная подсистема поддерживает 2 типа хранилища заданий:

– хранилище заданий в памяти: все данные о выполнении заданий хранятся в памяти экземпляра WildBoss Pro. При выключении сервера все данные о выполнении заданий теряются. В кластеризованной среде каждый Экземпляр сервера WildBoss Pro имеет собственное хранилище заданий в памяти, и обмен данными о выполнении заданий между экземплярами WildBoss Pro невозможен. Это хранилище заданий по умолчанию в пакетной подсистеме;

– хранилище заданий «jdbc»: все данные о выполнении заданий сохраняются в реляционной базе данных, доступ к которой осуществляется через «jdbc». В кластерной среде хранилище заданий «jdbc» может использоваться для обмена данными о выполнении заданий между экземплярами WildBoss Pro. Например, можно запустить выполнение задания в одном экземпляре, остановить и перезапустить его из другого экземпляра WildBoss Pro.

**Работа с большими хранилищами заданий**

В некоторых случаях, когда в хранилище заданий накапливается большое количество записей о выполнении заданий (например, сотни тысяч), это может отрицательно сказаться на времени развертывания приложения. Это относится в основном к реализациям постоянных хранилищ заданий, таким как хранилище заданий «jdbc».

Чтобы избежать накопления слишком большого количества записей о выполнении заданий, приложение может удалять старые выполнения с помощью JobOperator.abandon (длинный идентификатор выполнения) или использовать другие средства, такие как удаление таблиц базы данных.

Если избежать хранения большого количества записей о выполнении заданий невозможно, хранилища заданий можно настроить таким образом, чтобы ограничить количество возвращаемых ими записей о выполнении заданий, установив атрибут «executionrecords- limit». Если атрибут установлен, WildBoss Pro будет загружать только указанное максимальное количество выполнений заданий из резервного механизма хранения.

### 7.24.4 Дескрипторы развертывания

Нет описателей развертывания для настройки пакетной среды, определенной в JSR-352 спецификация. В WildBoss Pro вы можете использовать «jboss-all.xml» дескриптор развертывания для определения аспектов пакетной среды для вашего развертывания.

В дескрипторе развертывания jboss-all.xml вы можете определить именованное хранилище заданий, новое хранилище заданий и/или именованный пул потоков. Именованное хранилище заданий и именованный пул потоков — это ресурсы, определенные в пакетной подсистеме. В дескрипторе развертывания разрешено определять только именованный пул потоков.

Пример именованного хранилища заданий и пула потоков:

```
<jboss xmlns="urn:jboss:1.0">
  <batch xmlns="urn:jboss:domain:batch-jberet:2.0">
    <job-repository>
      <named name="batch-ds"/>
    </job-repository>
    <thread-pool name="deployment-thread-pool"/>
  </batch>
</jboss>
```

Пример нового хранилища заданий:

```
<jboss xmlns="urn:jboss:1.0">
  <batch xmlns="urn:jboss:domain:batch-jberet:2.0">
    <job-repository>
       <jdbc data-source="batch-ds"/>
    </job-repository>
  </batch>
</jboss>
```

### 7.24.5 Ресурсы для развертывания

Некоторые подсистемы в WildBoss Pro регистрируют ресурсы среды выполнения для развертываний. Пакетная подсистема регистрирует задания и их выполнение. Задания регистрируются с использованием имени задания, а не XML-имени задания. Выполнение регистрируется с использованием идентификатора выполнения.

Пакетное приложение на автономном сервере

```
[standalone@localhost:9990 /] /deployment=batch-jdbc-
chunk.war/subsystem=batchjberet:
read-resource(recursive=true,include-runtime=true)
{
  "outcome" => "success",
  "result" => {"job" => {
    "reader-3" => {
    "instance-count" => 1,
    "running-executions" => 0,
       "execution" => {"1" => {
          "batch-status" => "COMPLETED",
          "create-time" => "2015-08-07T15:37:06.416-0700",
          "end-time" => "2015-08-07T15:37:06.519-0700",
          "exit-status" => "COMPLETED",
          "instance-id" => 1L,
          "last-updated-time" => "2015-08-07T15:37:06.519-0700",
          "start-time" => "2015-08-07T15:37:06.425-0700"
       }}
    },
```

```
    "reader-5" => {
        "instance-count" => 0,
        "running-executions" => 0,
        "execution" => undefined
    }
  }}
}
```

Ресурс пакетной подсистемы в развертывании также выполняет 3 операции для взаимодействия с пакетными заданиями в выбранном развертывании. Есть операции запуска задания (start-job), остановки задания (stop-job) и перезапуска задания (restart-job). Ресурс выполнения также выполняет операции остановки задания (stop-job) и перезапуска задания (restart-job).

Пример начального задания

```
[standalone@localhost:9990 /] /deployment=batch-
chunk.war/subsystem=batchjberet:
start-job(job-xml-name=simple, properties={writer.sleep=5000})
{
 "outcome" => "success",
 "result" => 1L
}
```

Пример остановки задания

```
[standalone@localhost:9990 /] /deployment=batch-
chunk.war/subsystem=batch-jberet:stopjob(
```

Пример задания на перезапуск

```
[standalone@localhost:9990 /] /deployment=batch-
chunk.war/subsystem=batchjberet:
restart-job(execution-id=2)
{
  "outcome" => "success",
  "result" => 3L
}
```

Результат использования ресурса после 3 (трёх) выполнений

```
[standalone@localhost:9990 /] /deployment=batch-
chunk.war/subsystem=batch-jberet:readresource(
recursive=true, include-runtime=true)
{
  "outcome" => "success",
  "result" => {"job" => {"chunkPartition" => {
     "instance-count" => 2,
     "running-executions" => 0,
     "execution" => {
       "1" => {
         "batch-status" => "COMPLETED",
         "create-time" => "2015-08-07T15:41:55.504-0700",
         "end-time" => "2015-08-07T15:42:15.513-0700",
         "exit-status" => "COMPLETED",
         "instance-id" => 1L,
         "last-updated-time" => "2015-08-07T15:42:15.513-0700",
         "start-time" => "2015-08-07T15:41:55.504-0700"
       },
       "2" => {
```

```
            "batch-status" => "STOPPED",
            "create-time" => "2015-08-07T15:44:39.879-0700",
            "end-time" => "2015-08-07T15:44:54.882-0700",
            "exit-status" => "STOPPED",
            "instance-id" => 2L,
            "last-updated-time" => "2015-08-07T15:44:54.882-0700",
            "start-time" => "2015-08-07T15:44:39.879-0700"
        },
        "3" => {
            "batch-status" => "COMPLETED",
            "create-time" => "2015-08-07T15:45:48.162-0700",
            "end-time" => "2015-08-07T15:45:53.165-0700",
            "exit-status" => "COMPLETED",
            "instance-id" => 2L,
            "last-updated-time" => "2015-08-07T15:45:53.165-0700",
            "start-time" => "2015-08-07T15:45:48.163-0700"
        }
    }
  }}}
}
```

Совет профессионала

**Внимание:** можно отфильтровать задания по атрибуту ресурса выполнения с помощью операции запроса.

Просмотр всех остановленных заданий

```
/deployment=batch-chunk.war/subsystem=batchjberet/
job=*/execution=*:query(where=["batch-status", "STOPPED"])
```

Как и в случае со всеми операциями, вы можете просмотреть подробную информацию об операции, используя операцию описания «:read-operationdescription».

Заполнение вкладки.

**Внимание:** не забывайте, что в CLI есть вкладка «completion», которая завершает операции и атрибуты (аргументы) для операций.

Пример описания операции запуска задания

```
[standalone@localhost:9990 /] /deployment=batch-
chunk.war/subsystem=batch-jberet:readoperation-description(name=start-
job)
{
  "outcome" => "success",
  "result" => {
     "operation-name" => "start-job",
     "description" => "Starts a batch job.",
     "request-properties" => {
       "job-xml-name" => {
         "type" => STRING,
         "description" => "The name of the job XML file to use when
         starting
         the job.",
         "expressions-allowed" => false,
         "required" => true,
         "nillable" => false,
         "min-length" => 1L,
         "max-length" => 2147483647L
       },
       "properties" => {
         "type" => OBJECT,
```

```
            "description" => "Optional properties to use when starting
            the batch
            job.",
            "expressions-allowed" => false,
            "required" => false,
            "nillable" => true,
            "value-type" => STRING
          }
        },
        "reply-properties" => {"type" => LONG},
        "read-only" => false,
        "runtime-only" => true
    }
}
```

## 7.25 Конфигурирование Jakarta Server Faces

Конфигурирование Jakarta Server Faces осуществляется подсистемой Jakarta Server Faces. Подсистема Jakarta Server Faces позволяет использовать несколько серверов Jakarta Server Реализации Faces должны быть установлены на том же сервере WildBoss Pro. В частности, может быть установлена любая версия «Mojarra» или «MyFaces», реализующая спецификации уровня 2.1 или выше. Для каждой реализации Jakarta Server Faces необходимо создать новый слот в «com.sun.jsf-impl», «javax.faces.api» и «org.jboss.as.jsfinjection». Когда подсистема Jakarta Server Faces запускается, она сканирует путь к модулю, чтобы найти все установленные реализации Jakarta Server Faces. Реализация Jakarta Server Faces по умолчанию, которую должен использовать WildBoss Pro, определяется атрибутом «default-jsf-impl-slot».

### 7.25.1 Установка нового сервера Jakarta требует выполнения вручную

Новая реализация Jakarta Server Faces может быть установлена вручную следующим образом:

1) добавьте слот для модуля для новой версии JAR Jakarta Server Faces implementation:

– создайте следующую структуру каталогов в каталоге «WILDBOSS PRO_HOME/modules: WILDBOSS PRO_HOME/modules/com/sun/jsf-impl/<JSF_IMPL_NAME>-<JSF_VERSION>».

Например, для Mojarra 2.2.11 указанный выше путь будет выглядеть следующим образом: «WILDBOSS PRO_HOME/modules/com/sun/jsf-impl/mojarra-2.2.11»;

– поместите JAR-файл реализации Jakarta Server Faces в подкаталог <JSF_IMPL_NAME>-<JSF_VERSION>. В том же подкаталоге добавьте файл «module.xml», аналогичный примерам шаблонов «Mojarra» или «MyFaces». Измените корневой путь к ресурсу (resource-root-path) на имя вашего сервера Jakarta Server Faces реализации JAR и введите соответствующие значения «$\{ jsf-impl-name}» и «$\{ jsf-version}»;

2) добавьте слот для модуля для нового JAR-интерфейса Jakarta Server Faces API:

– создайте следующую структуру каталогов в каталоге «WILDBOSS PRO_HOME/modules: WILDBOSS PRO_HOME/modules/javax/faces/api/<JSF_IMPL_NAME>-<JSF_VERSION>»;

– поместите JAR-файл JSF API в подкаталог <JSF_IMPL_NAME>-<JSF_VERSION>. В этот же подкаталог добавьте файл «module.xml», аналогичный примерам шаблонов «Mojarra» или «MyFaces». Измените корневой путь к ресурсу на имя вашего сервера Jakarta Server Faces API JAR и введите соответствующие значения «$\{ jsf-impl-name}» и «$\{ jsf-version}»;

3) добавьте модульный слот для Jakarta Server Faces injection JAR

– создайте следующую структуру каталогов в каталоге «WILDBOSS PRO_HOME/modules: WILDBOSS PRO_HOME/modules/org/jboss/as/jsf-injection/<JSF_IMPL_NAME>-<JSF_VERSION>»;

– скопируйте JAR-файл «WildBoss Pro-jsf-injection» и файл «weld-core-jsf» из «WILDBOSS PRO_HOME/modules/system/layers/base/org/jboss/as/jsf-injection/main» в подкаталог <JSF_IMPL_NAME>-<JSF_VERSION>;

– в подкаталоге <JSF_IMPL_NAME>-<JSF_VERSION> добавьте «module.xml» файл, аналогичный примерам шаблонов «Mojarra» или «MyFaces», и введите соответствующие значения «$\{ jsf-impl-name}», «$\{ jsf-версия}», «$\{ version.jboss.as}» и «$\{ version.weld.core}». (Эти последние два заполнителя зависят от версий файлов «WildBoss Pro-jsf-injection» и «weld-core-jsf», которые были скопированы на предыдущем шаге);

4) только для MyFaces - добавьте модуль для «commons-digester JAR»:

– создайте следующую структуру каталогов в каталоге «WILDBOSS PRO_HOME/modules: WILDBOSS PRO_HOME/modules/org/apache/commons/digestor/main»;

– Поместите «commons-digester» JAR в «WILDBOSS PRO_HOME/modules/org/apache/commons/digester/main». В подкаталог «main» добавить файл «module.xml», аналогичный данному шаблону. Введите соответствующее значение «$\{ version.commonsdigester}»;

5) запустите сервер:

После запуска сервера можно использовать следующую команду CLI для проверки того, что данная новая реализация Jakarta Server Faces была успешно установлена. В выходных данных этой команды должна появиться новая реализация Jakarta Server Faces.

```
[standalone@localhost:9990 /] /subsystem=jsf:list-active-jsf-impls()
```

### 7.25.2 Необходимые изменения сервера Jakarta по умолчанию

Следующая команда CLI может быть использована для того, чтобы сделать недавно установленную реализацию Jakarta Server Faces реализацией Jakarta Server Faces по умолчанию, используемой WildBoss Pro:

```
/subsystem=jsf/:write-attribute(name=default-jsf-impl-
slot,value=<JSF_IMPL_NAME>-<JSF_VERSION>)
```

Для вступления этого изменения в силу потребуется перезагрузка сервера.

### 7.25.3 Настройка приложения Jakarta Server Faces для использования приложения Jakarta, отличного от используемой по умолчанию реализации Server Face

Приложение Jakarta Server Faces можно настроить для использования установленной реализации Jakarta Server Faces, которая не является реализацией по умолчанию, добавив параметр контекста «org.jboss.jbossfaces.JSF_CONFIG_NAME» в его файл «web.xml». Например, чтобы указать, что приложение Jakarta Server Faces должно использовать MyFaces 2.2.12 (при условии, что на сервере установлена версия MyFaces 2.2.12), необходимо добавить следующий параметр контекста:

```
<context-param>
   <param-name>org.jboss.jbossfaces.JSF_CONFIG_NAME</param-name>
   <param-value>myfaces-2.2.12</param-value>
</context-param>
```

Если в приложении Jakarta Server Faces не указан этот параметр контекста, то для этого приложения будет использоваться реализация для приложения Jakarta Server Faces.

### 7.25.4 Запрет объявлений типа «DOCTYPE»

Следующие команды CLI можно использовать для запрета объявлений «DOCTYPE» в развертываниях Jakarta Server Faces:

```
/subsystem=jsf:write-attribute(name=disallow-doctype-decl, value=true)
reload
```

Этот параметр можно переопределить для конкретного Jakarta Server Faces развертывания, «com.sun.faces.disallowDoctypeDecl» параметр контекста для параметров развертывания файла «web.xml»:

```
<context-param>
   <param-name>com.sun.faces.disallowDoctypeDecl</param-name>
   <param-value>false</param-value>
</context-param>
```

## 7.26 Конфигурация подсистемы JMX

Подсистема JMX регистрирует службу в конечной точке удаленного взаимодействия, чтобы можно было получить удаленный доступ к JMX через открытый соединитель удаленного взаимодействия.

По умолчанию этот параметр включен в автономном режиме и доступен через порт 9990, но в режиме домена он выключен, поэтому его необходимо включить - в режиме домена этот порт будет являться портом соединителя удаленного взаимодействия для экземпляра WildBoss Pro, который будет контролироваться.

Чтобы использовать коннектор, вы можете получить к нему доступ стандартным способом, используя «service:jmx» URL:

```
import javax.management.MBeanServerConnection;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import javax.management.remote.JMXServiceURL;
public class JMXExample {
   public static void main(String[] args) throws Exception {
        //Get a connection to the WildBoss Pro MBean server on localhost
        String host = "localhost";
        int port = 9990; // management-web port
        String urlString =
           System.getProperty("jmx.service.url","service:jmx:remote+http
           ://" + host +":" + port);
        JMXServiceURL serviceURL = new JMXServiceURL(urlString);
        JMXConnector jmxConnector =
        JMXConnectorFactory.connect(serviceURL, null);
        MBeanServerConnection connection =
        jmxConnector.getMBeanServerConnection();
        //Invoke on the WildBoss Pro MBean server
        int count = connection.getMBeanCount();
        System.out.println(count);
        jmxConnector.close();
   }
}
```

Также необходимо указать путь к классу при запуске приведенного выше примера. Следующий сценарий описывает Linux. Если ваша среда сильно отличается, вставьте свой скрипт, когда он у вас заработает.

```
!/bin/bash
# specify your WildBoss Pro folder +
export YOUR_JBOSS_HOME=~/WildBoss Pro
java -classpath $YOUR_JBOSS_HOME/bin/client/jboss-client.jar:./
JMXExample
```

Также можно подключиться с помощью «jconsole».

**Внимание:** если используется «jconsole», нужно использовать скрипты «jconsole.sh» и «jconsole.bat», включенные в каталог «/bin» дистрибутива WildBoss Pro, поскольку они задают путь к классу, необходимый для подключения через удаленное взаимодействие.

В дополнение к стандартным MBeans JVM, сервер WildBoss Pro MBean содержит следующие MBeans:

Таблица 11 - Дополнения к стандартным MBeans JVM

| Имя объекта JMX | Описание |
|---|---|
| jboss.msc:type=container,name=jboss-as | Предоставляет доступ к операциям управления в JBoss Modular Service Container, который является платформой внедрения зависимостей, лежащей в основе WildBoss Pro. Это полезно для отладки проблем с зависимостями, например, если вы интегрируете свои собственные подсистемы, поскольку позволяет выполнять операции по удалению всех служб и их текущих состояний |
| jboss.naming:type=JNDIView | Показывает, что связано в JNDI |
| jboss.modules:type=ModuleLoader,name=* | Эта коллекция MBeans предоставляет доступ к операциям управления на уровне загрузки классов модулей JBoss. Это полезно для отладки проблем с зависимостями, возникающих из-за отсутствия зависимостей модулей |

### 7.26.1 Ведение журнала аудита

Ведение журнала аудита для сервера JMX MBean, управляемого подсистемой JMX. Ресурс находится по адресу «/subsystem=jmx/configuration=audit-log», и его атрибуты аналогичны атрибутам, указанным для «/core-service=management/access=audit/logger=audit-log» в журнале аудита.

Таблица 12 - Атрибуты

| Атрибут | Описание |
|---|---|
| enabled | значение «true», чтобы включить ведение журнала операций JMX |

| Атрибут | Описание |
|---|---|
| log-boot | значение «true» для регистрации операций JMX при загрузке сервера, значение «false» в противном случае |
| log-read-only | Если значение «true», то все операции будут записываться в журнал аудита, если значение «false» , то будут записываться только операции, которые изменяют модель |

Затем, обработчики, которые используются для регистрации операций управления, настраиваются как «handler=*» дочерние элементы регистратора. Эти обработчики и средства их форматирования определены в глобальном разделе «/coreservice=management/access=audit», упомянутом в журнале аудита.

*Средство форматирования JSON* - используется тот же формат JSON, что и описанный в разделе «Ведение журнала аудита». Однако записи для MBean Вызовов сервера содержат поля, немного отличающиеся от тех, которые регистрируются на базовом уровне управления.

```
2013-08-29 18:26:29 - {
   "type" : "jmx",
   "r/o" : false,
   "booting" : false,
   "version" : "10.0.0.Final",
   "user" : "$local",
   "domainUUID" : null,
   "access" : "JMX",
   "remote-address" : "127.0.0.1/127.0.0.1",
   "method" : "invoke",
   "sig" : [
      "javax.management.ObjectName",
      "java.lang.String",
      "[Ljava.lang.Object;",
      "[Ljava.lang.String;"
   ],
   "params" : [
      "java.lang:type=Threading",
      "getThreadInfo",
      "[Ljava.lang.Object;@5e6c33c",
      "[Ljava.lang.String;@4b681c69"
   ]
}
```

Он включает в себя необязательную временную метку, а затем следующую информацию в записи «json».

Таблица 13

| Имя поля | Описание |
|---|---|
| type | это будет иметь значение «jmx», означающее, что оно поступает из подсистемы «jmx» |
| r/o | «true», если операция влияет на MBean(ы) только для чтения |
| booting | «true», если операция была выполнена во время процесса загрузки, «false», если она была выполнена после запуска сервера |
| version | номер версии экземпляра WildBoss Pro |
| user | имя пользователя, прошедшего проверку подлинности |

| Имя поля | Описание |
|---|---|
| domainUUID | в настоящее время это поле не заполнено для операций JMX |
| access | это может иметь одно из следующих значений: «*NATIVE» - операция была выполнена через собственный интерфейс управления, например «CLI*HTTP» - Операция была выполнена через HTTP-интерфейс домена, например, консоль администратора «*JMX» - Операция была выполнена через подсистему JMX. Как настроить ведение журнала аудита для JMX, смотрите в разделе JMX |
| remote-address | адрес клиента, выполняющего эту операцию |
| method | имя вызываемого метода «MBeanServer» |
| sig | сигнатура вызываемого метода «called MBeanServer» |
| params | фактические параметры передаются методу «MBeanServer», для каждого параметра вызывается простой Object.toString() |
| error | если вызов метода MBeanServer привел к ошибке, это поле будет заполнено значением Throwable.GetMessage() |

### 7.27 Конфигурация сканера для развертывания

Сканер развертывания используется только в автономном режиме. Его задача - отслеживать каталог на наличие новых файлов и развертывать эти файлы. Его можно найти в «standalone.xml»:

```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:2.0">
  <deployment-scanner scan-interval="5000"
           relative-to="jboss.server.base.dir" path="deployments" />
</subsystem>
```

Можно указать дополнительные записи сканера развертывания для поиска развертываний в большем количестве местоположений. Показанная конфигурация будет сканировать каталог «JBOSS_HOME/standalone/deployments» каждые пять секунд. Модель среды выполнения показана ниже и использует значения по умолчанию для атрибутов (таблица 14), не указанных в «xml»:

Таблица 14 - Атрибуты

| Имя | Тип | Описание |
|---|---|---|
| name | STRING | Имя сканера. Если оно не указано, используется значение по умолчанию |
| path | STRING | Фактический путь к файловой системе, который необходимо проверить. Рассматривается как абсолютный путь, если не указан атрибут «relative-to», и в этом случае значение рассматривается как относительное к этому пути |
| relative-to | STRING | Ссылка на путь к файловой системе, определенный в разделе «Пути» конфигурации сервера, или на одно из системных свойств, указанных при запуске. В приведенном выше примере файл «jboss.server.base.dir» преобразуется в «JBOSS_HOME/standalone» |
| scan-enabled | BOOLEAN | Если включено истинное сканирование |

| Имя | Тип | Описание |
|---|---|---|
| scan-interval | INT | Периодический интервал в миллисекундах, с которым хранилище следует проверять на наличие изменений. Значение меньше 1 означает, что хранилище следует проверять только при первоначальном запуске |
| auto-deploy-zipped | BOOLEAN | Определяет, должен ли сканер автоматически развертывать архивированное содержимое для развертывания, не требуя от пользователя добавления файла маркера «.dodeploy» |
| auto-deploy-exploded | BOOLEAN | Определяет, должен ли сканер автоматически развертывать содержимое разнесенного развертывания, не требуя от пользователя добавления файла маркера «.dodeploy». Установка значения «true» не рекомендуется ни для чего, кроме базовых сценариев разработки, поскольку невозможно гарантировать, что развертывание не произойдет в процессе внесения изменений в содержимое |
| auto-deploy-xml | BOOLEAN | Определяет, должно ли содержимое XML автоматически развертываться сканером без использования файла маркера «.dodeploy» |
| deployment-timeout | LONG | Время ожидания в секундах, которое требуется для выполнения развертывания перед отменой. Значение по умолчанию - 60 секунд |

Сканеры развертывания можно добавить, изменив «standalone.xml» перед запуском сервера, или их можно добавлять и удалять во время выполнения с помощью интерфейса командной строки.

```
[standalone@localhost:9990 /] /subsystem=deployment-
scanner/scanner=new:add(scaninterval=10000,relative-
to="jboss.server.base.dir",path="other-deployments")
{"outcome" => "success"}
[standalone@localhost:9990 /] /subsystem=deployment-
scanner/scanner=new:remove
{"outcome" => "success"}
```

Вы также можете изменить атрибуты во время выполнения, так что, например, чтобы отключить сканирование, вы можете сделать.

```
[standalone@localhost:9990 /] /subsystem=deployment-
scanner/scanner=default:writeattribute(name="scan-
enabled",value=false)
{"outcome" => "success"}
[standalone@localhost:9990 /] /subsystem=deployment-
scanner:readresource(
recursive=true)
{
   "outcome" => "success",
   "result" => {"scanner" => {"default" => {
     "auto-deploy-exploded" => false,
     "auto-deploy-zipped" => true,
     "deployment-timeout" => 60L,
```

```
    "name" => "default",
    "path" => "deployments",
    "relative-to" => "jboss.server.base.dir",
    "scan-enabled" => false,
    "scan-interval" => 5000
  }}}
}
```

### 7.28 Конфигурация основной подсистемы управления

Основная подсистема управления состоит из служб, используемых для управления сервером или мониторинга его состояния.

Конфигурация основной подсистемы управления может использоваться для:

– зарегистрируйте прослушиватель для событий жизненного цикла сервера;

– список последних изменений конфигурации на сервере.

#### 7.28.1 Прослушиватель жизненного цикла

Вы можете создать реализацию «*org.WildBoss Pro.extension.core.management.client.ProcessStateListener*», который будет получать уведомления об изменениях состояния запуска и конфигурации среды выполнения, что позволит разработчику реагировать на эти изменения.

Чтобы использовать эту функцию, вам необходимо создать свой собственный модуль, затем настроить и развернуть его с помощью основной подсистемы управления.

Например, давайте создадим простой прослушиватель:

```
public class SimpleListener implements ProcessStateListener {
 private File file;
 private FileWriter fileWriter;
 private ProcessStateListenerInitParameters parameters;
 @Override
 public void init(ProcessStateListenerInitParameters parameters) {
  this.parameters = parameters;
  this.file = new File(parameters.getInitProperties().get("file"));
  try {
    fileWriter = new FileWriter(file, true);
    } catch (IOException e) {
    e.printStackTrace();
  }
 }
 @Override
 public void cleanup() {
   try {
     fileWriter.close();
     } catch (IOException e) {
     e.printStackTrace();
     } finally {
     fileWriter = null;
   }
 }
 @Override
 public void
 runtimeConfigurationStateChanged(RuntimeConfigurationStateChangeEvent
 evt) {
   try {
```

```
      fileWriter.write(String.format("%s %s %s %s\n",
      parameters.getProcessType
      (), parameters.getRunningMode(), evt.getOldState(),
      evt.getNewState()));
      } catch (IOException e) {
      e.printStackTrace();
    }
  }
  @Override
  public void runningStateChanged(RunningStateChangeEvent evt) {
    try {
      fileWriter.write(String.format("%s %s %s %s\n",
      parameters.getProcessType
      (), parameters.getRunningMode(), evt.getOldState(),
      evt.getNewState()));
      } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Чтобы скомпилировать его, нужно использовать модуль «org.WildBoss Pro.core:WildBoss Pro-core-management-client maven». Теперь нужно добавить этот модуль к модулям WildBoss Pro:

```
module add --name=org.simple.lifecycle.events.listener
--dependencies=org.WildBoss Pro.extension.core-management-client
--resources=/home/ehsavoie/dev/demo/simple-listener/target/simple-
process-state-listener.jar
```

Теперь можно зарегистрироваться или прослушать:

```
/subsystem=core-management/process-state-
listener=simplelistener:add(class=org.simple.lifecycle.events.listener.
SimpleListener,
module=org.simple.lifecycle.events.listener,properties={file=/home/Wild
Boss Pro/tmp/events.txt})
```

### 7.28.2 Изменения конфигурации

Можно использовать основную подсистему управления, чтобы включить и настроить сохраняемую в памяти историю последних изменений конфигурации.

Например, чтобы отслеживать последние 5 изменений конфигурации, активируем это:

```
/subsystem=core-management/service=configuration-changes:add(max-
history=5)
```

Теперь мы можем перечислить последние изменения конфигурации:

```
/subsystem=core-management/service=configuration-changes:list-
changes()
{
  "outcome" => "success",
  "result" => [{
```

```
    "operation-date" => "2016-12-05T11:05:12.867Z",
    "access-mechanism" => "NATIVE",
    "remote-address" => "/127.0.0.1",
    "outcome" => "success",
    "operations" => [{
      "address" => [
        ("subsystem" => "core-management"),
        ("service" => "configuration-changes")
      ],
      "operation" => "add",
      "max-history" => 5,
      "operation-headers" => {
         "caller-type" => "user",
         "access-mechanism" => "NATIVE"
      }
    }]
  }]
}
```

### 7.28.3 Конфигурация подсистемы JAXRS

Подсистема «jaxrs» представляет веб-сервисы Jakarta RESTful. Реализацией является RESTEasy.

*Требуется расширение*: требуемое расширение находится в модуле «org.jboss.as.jaxrs». В большинстве случаев расширение должно присутствовать. Однако, если это не так, можно добавить его с помощью CLI.

```
/extension=org.jboss.as.jaxrs:add
```

При этом добавляется следующая запись конфигурации:

```
<extension module="org.jboss.as.jaxrs"/>
```

*Пример базовой конфигурации подсистемы*: по умолчанию подсистема «jaxrs» пуста, что приводит к использованию значений конфигурации по умолчанию.

Это можно изменить с помощью клиента управления, такого как CLI или веб-консоль. Можно получить подробную информацию о «Параметрах конфигурации» в разделе 3.5 «Руководства пользователя RESTEasy».

Пример настройки подсистемы с помощью интерфейса командной строки:

```
/subsystem=jaxrs:write-attribute(name=resteasy-add-charset,
value=true)
/subsystem=jaxrs:write-attribute(name=resteasy-gzip-max-input,
value=17)
/subsystem=jaxrs:write-attribute(name=resteasy-gzip-max-input,
value=17)
/subsystem=jaxrs:write-attribute(name=resteasy-jndi-resources,
value=["java:global/jaxrsnoap/EJB_Resource1",
"java:global/jaxrsnoap/EJB_Resource2"])
/subsystem=jaxrs:write-attribute(name=resteasy-language-mappings,
value={"es"="es","fr"="fr", "en"="en-US"})
/subsystem=jaxrs:write-attribute(name=resteasy-media-type-param-
mapping, value=mt)
```

```
/subsystem=jaxrs:write-attribute(name=resteasy-providers,
value=["com.bluemonkey.reader", "com.bluemonkey.writer"])
```

При этом создается следующая конфигурация XML.

```
<subsystem xmlns="urn:jboss:domain:jaxrs:2.0">
   <resteasy-add-charset>true</resteasy-add-charset>
   <resteasy-gzip-max-input>17</resteasy-gzip-max-input>
   <resteasy-jndi-resources>
      <jndi>
         java:global/jaxrsnoap/EJB_Resource1
      </jndi>
      <jndi>
         java:global/jaxrsnoap/EJB_Resource2
      </jndi>
   </resteasy-jndi-resources>
   <resteasy-language-mappings>
      <entry key="es">
         es
      </entry>
      <entry key="fr">
         fr
      </entry>
      <entry key="en">
         en-US
      </entry>
   </resteasy-language-mappings>
   <resteasy-media-type-param-mapping>mt</resteasy-media-type-param-
   mapping>
   <resteasy-providers>
      <class>
         com.bluemonkey.reader
      </class>
      <class>
         com.bluemonkey.writer
      </class>
   </resteasy-providers>
</subsystem>
```

**Внимание:** использование дефисов является обычным делом. Перед передачей параметров в RESTEasy дефисы заменяются точками, чтобы они соответствовали именам параметров RESTEasy.

Подробное описание различных параметров приведено в «Руководстве пользователя RESTEasy».

**Внимание:** важно понимать, что эти параметры являются глобальными. То есть они применимы ко всем развертываниям. Поскольку эти параметры являются глобальными, классы, упомянутые в «resteasy.providers» и «resteasy.disable.поставщики» должны быть доступны для всех развертываний. Таким образом, на практике они предназначены для включения или выключения Поставщики услуг RESTEasy. Обратите внимание, что они могут использоваться совместно с «resteasy-usebuildin- providers» для настройки набора доступных поставщиков.

**Внимание:** другим важным фактом является то, что после изменения параметров через какой-либо интерфейс управления эти изменения требуют повторного развертывания всех ранее развернутых приложений.

**Внимание:** RESTEasy представила новую обработку исключения «javax.ws.rs.WebApplicationException», создаваемого клиентом Jakarta REST или MicroProfile

REST клиент, работающий внутри RESTful ресурса, в который встроен «javax.ws.rs.core». Ответ «очищается» перед отправкой, чтобы предотвратить риск утечки информации от третьей стороны. Исходное поведение можно восстановить, установив для параметра «resteasy.original.webapplicationexception.behavior» значение «true». Смотрите раздел Руководства пользователя RESTEasy «Исключения из веб-приложений Resteasy» для получения дополнительной информации.

## 7.29 Конфигурация клиентской подсистемы Elytron OpenID Connect

Возможность защищать приложения с помощью OpenID Connect обеспечивается клиентской подсистемой «elytron-oidc».

### 7.29.1 Подсистема

Подсистема «elytron-oidc-client» включена в конфигурацию по умолчанию. Если подсистема отсутствует, ее можно добавить, используя следующие команды интерфейса командной строки.

```
[standalone@localhost:9990 /] /extension=org.WildBoss
Pro.extension.elytron-oidc-client:add
[standalone@localhost:9990 /] /subsystem=elytron-oidc-client:add
[standalone@localhost:9990 /] reload
```

### 7.29.2 Конфигурация

По умолчанию клиентская подсистема «elytron-oidc» не содержит настроенных ресурсов или атрибутов.

Конфигурация, необходимая для защиты приложения с помощью OpenID Connect, может быть предоставлена либо в самом приложении, либо в клиентской подсистеме «elytron-oidc».

*Конфигурация развертывания* - конфигурация, необходимая для обеспечения безопасности приложения с помощью OpenID Connect, может быть указана при развертывании.

Первым шагом является создание файла конфигурации «oidc.json» в каталоге WEB-INF приложения. Второй шаг - установить для метода аутентификации значение OIDC в файле приложения «web.xml».

Вот пример файла конфигурации «oidc.json»:

```
{
  "client-id" : "customer-portal",
  "provider-url" : "http://localhost:8180/auth/realms/demo",
  "ssl-required" : "external",
  "use-resource-role-mappings" : false,
  "enable-cors" : true,
  "cors-max-age" : 1000,
  "cors-allowed-methods" : "POST, PUT, DELETE, GET",
  "cors-exposed-headers" : "WWW-Authenticate, My-custom-exposed-
  Header",
  "enable-basic-auth" : false,
  "expose-token" : true,
  "verify-token-audience" : true,
   "credentials" : {
      "secret" : "234234-234234-234234"
   },
```

```
  "connection-pool-size" : 20,
  "socket-timeout-millis": 5000,
  "connection-timeout-millis": 6000,
  "connection-ttl-millis": 500,
  "disable-trust-manager": false,
  "allow-any-hostname" : false,
  "truststore" : "path/to/truststore.jks",
  "truststore-password" : "geheim",
  "client-keystore" : "path/to/client-keystore.jks",
  "client-keystore-password" : "geheim",
  "client-key-password" : "geheim",
  "token-minimum-time-to-live" : 10,
  "min-time-between-jwks-requests" : 10,
  "public-key-cache-ttl": 86400,
  "redirect-rewrite-rules" : {
  "^/wsmaster/api/(.*)$" : "/api/$1"
  }
}
```

*Конфигурация подсистемы* - вместо добавления конфигурации в ваше развертывание для обеспечения его безопасности с помощью OpenID Connect, как описано в предыдущем разделе, другим вариантом является добавление конфигурации в подсистему «elytron-oidc-client».

В следующем примере показано, как добавить конфигурацию в подсистему «elytron-oidc-client».

```
<subsystem xmlns="urn:WildBoss Pro:elytron-oidc-client:1.0">
 <secure-deployment name="DEPLOYMENT_RUNTIME_NAME.war">
   <client-id>customer-portal</client-id>
   <provider-url>http://localhost:8180/auth/realms/demo</provider-
   url>
   <ssl-required>external</ssl-required>
   <credential name="secret" secret="0aa31d98-e0aa-404c-b6e0-
   e771dba1e798" />
 </secure-deployment
</subsystem>
```

Ресурс безопасного развертывания позволяет вам предоставить конфигурацию для конкретного развертывания. В приведенном выше примере ресурс безопасного развертывания предоставляет конфигурацию, которая должна использоваться для развертывания «DEPLOYMENT_RUNTIME_NAME.war», где «DEPLOYMENT_RUNTIME_NAME» соответствует имени среды выполнения для развертывания.

Различные параметры конфигурации, которые могут быть указаны в конфигурации безопасного развертывания , соответствуют тем же параметрам, которые могут быть указаны в конфигурации «oidc.json», описанной в предыдущем разделе.

Если у вас есть несколько приложений, защищаемых с помощью одного и того же поставщика OpenID, конфигурация поставщика может быть определена отдельно, как показано в примере ниже:

```
<subsystem xmlns="urn:WildBoss Pro:elytron-oidc-client:1.0">
  <provider name="keycloak">
    <provider-url>http://localhost:8080/auth/realms/demo</provider-
    url>
    <ssl-required>external</ssl-required>
  </realm>
  <secure-deployment name="customer-portal.war">
    <provider>keycloak</provider>
    <client-id>customer-portal</client-id>
    <credential name="secret" secret="0aa31d98-e0aa-404c-b6e0-
    e771dba1e798" />
  </secure-deployment>
  <secure-deployment name="product-portal.war">
    <provider>keycloak</provider>
    <client-id>product-portal</client-id>
    <credential name="secret" secret="0aa31d98-e0aa-404c-b6e0-
    e771dba1e798" />
  </secure-deployment>
</subsystem>
```

*Активация* - подсистема «elytron-oidc-client» будет сканировать развертывания, чтобы определить, требуется ли механизм аутентификации OIDC для каких-либо веб-компонентов (т.е. для каждого развертывания подсистема будет определять, была ли найдена конфигурация OIDC в рамках развертывания или есть ли конфигурация OIDC для развертывания в конфигурации подсистемы). Если подсистема обнаружит, что механизм OIDC действительно требуется, она автоматически активирует механизм аутентификации. В противном случае активация не произойдет, и развертывание продолжится в обычном режиме.

### 7.29.3 Виртуальная безопасность

Целью использования OpenID Connect является проверка личности пользователя на основе проверки подлинности, выполненной поставщиком OpenID. По этой причине развертывания OpenID Connect не зависят от ресурсов домена безопасности, которые были определены в подсистеме Elytron, в отличие от традиционных развертываний. Вместо этого подсистема «elytron-oidc-client» автоматически создаст и будет использовать свой собственный виртуальный домен безопасности в рамках развертывания. Дополнительная управляемая настройка не требуется.

### 7.29.4 Поставщики OpenID

Атрибут «provider-url» в конфигурации «oidc.json» и в конфигурации «elytron-oidc-client» подсистемы позволяет вам указать URL-адрес поставщика OpenID, который вы хотели бы использовать. Для В версии WildBoss Pro 25 подсистема «elytron-oidc-client» была протестирована с помощью поставщика Keycloak OpenID. Другие поставщики OpenID OpenIDs еще не прошли тщательного тестирования, поэтому их использование пока следует рассматривать как экспериментальное и не следует использовать в рабочей среде. Надлежащая поддержка других поставщиков OpenIDs будет добавлена в следующем выпуске WildBoss Pro.

### 7.29.5 Поддержка нескольких арендаторов

В некоторых случаях может оказаться желательным защитить приложение, используя несколько файлов конфигурации «oidc. Json». Например, в зависимости от запроса может потребоваться использовать другой файл «oidc.json», чтобы аутентифицировать пользователей из нескольких областей Keycloak. Подсистема «elytron-oidc-client» позволяет использовать пользовательский преобразователь конфигурации, чтобы вы могли определить, какой файл конфигурации использовать для каждого запроса.

Чтобы использовать функцию множественной аренды, вам необходимо создать класс, который реализует интерфейс «org.WildBoss Pro.security.http.oidc.OidcClientConfigurationResolver», как показано в примере ниже:

```
package example;
import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import org.WildBoss Pro.security.http.oidc.OidcClientConfiguration;
import org.WildBoss
Pro.security.http.oidc.OidcClientConfigurationBuilder;
import org.WildBoss
Pro.security.http.oidc.OidcClientConfigurationResolver;
import org.WildBoss Pro.security.http.oidc.OidcHttpFacade;
public class MyCustomConfigResolver implements
OidcClientConfigurationResolver {
 private final Map<String, OidcClientConfiguration> cache = new
 ConcurrentHashMap<>();
 @Override
 public OidcClientConfiguration resolve(OidcHttpFacade.Request
 request) {
   String path = request.getURI();
   String realm = ... // determine which Keycloak realm to use based
   on the
   request path
   OidcClientConfiguration clientConfiguration = cache.get(realm);
   if (clientConfiguration == null) {
     InputStream is = getClass().getResourceAsStream("/oidc-" + realm
     + ".json
     "); // config to use based on the realm
     clientConfiguration = OidcClientConfigurationBuilder.build(is);
     cache.put(realm, clientConfiguration);
   }
   return clientConfiguration;
 }
}
```

После того, как вы создали свой «OidcClientConfigurationResolver», вы можете указать, что хотите использовать свой пользовательский преобразователь конфигурации, установив контекстный параметр «oidc.config.resolver» в вашем приложении «web.xml» файл, как показано в примере ниже:

```
<web-app>
   ...
   <context-param>
      <param-name>oidc.config.resolver</param-name>
      <param-value>example.MyCustomConfigResolver</param-value>
   </context-param>
   ...
</web-app>
```

### 7.30 Простая настройка подсистем

Следующие подсистемы в настоящее время не имеют конфигурации, выходящей за рамки их корневого элемента в конфигурации.

```
<subsystem xmlns="urn:jboss:domain:jdr:1.0"/>
<subsystem xmlns="urn:jboss:domain:pojo:1.0"/>
<subsystem xmlns="urn:jboss:domain:sar:1.0"/>
```

Наличие каждого из них включает в себя определенную функциональность (таблица 15).

Таблица 15

| Имя | Описание |
|---|---|
| jdr | Позволяет собирать диагностические данные для использования при удаленном анализе условий возникновения ошибок. Хотя данные представлены в простом формате и могут быть полезны любому пользователю, в первую очередь они будут полезны подписчикам JBoss EAP, которые будут предоставлять данные Red. При обращении в службу поддержки сообщите об этом |
| pojo | Позволяет развертывать приложения, содержащие службы JBoss Microcontainer, которые поддерживались предыдущими версиями JBoss Application Server |
| sar | Позволяет развертывать архивы «.SAR», содержащие службы «MBean», которые поддерживались предыдущими версиями JBoss Application Server |

# 8 Настройка домена

Чтобы запустить группу серверов в качестве управляемого домена, вам необходимо настроить как контроллер домена, так и каждый хост, который присоединяется к домену. В этом разделе описывается конфигурация сети для компонентов домена и контроллера хоста. Для получения дополнительной информации пользователям рекомендуется ознакомиться с разделами «Режимы работы» и «Файлы конфигурации».

## 8.1 Конфигурация контроллера домена

Контроллер домена является центральным органом управления управляемым доменом. Для настройки контроллера домена требуется выполнить два шага:

– хост должен быть настроен так, чтобы он выступал в качестве контроллера домена для всего домена;

– хост должен предоставить адресуемую привязку интерфейса управления, чтобы управляемые хосты могли взаимодействовать с ним.

*Примеры IP-адресов.*

**Внимание:** в этом примере контроллер домена использует 192.168.0.101 и хост-контроллер 192.168.0.10

Настройка хоста для выполнения функций контроллера домена выполняется с помощью объявления «domain-controller» в «host.xml». Если оно содержит элемент «local/», то этот хост станет контроллером домена:

```
<domain-controller>
  <local/>
</domain-controller>
```

(Смотри domain/configuration/host.xml)

Хост, выполняющий функции контроллера домена, должен предоставлять интерфейс управления по адресу, доступному для других хостов в домене. Предоставление интерфейса управления HTTP(S) не требуется, но рекомендуется, поскольку это позволяет работать консоли администрирования:

```
<management-interfaces>
  <native-interface sasl-authentication-factory="management-sasl-
  authentication">
    <socket interface="management" port="9999"/>
  </native-interface>
  <http-interface http-authentication-factory="management-http-
  authentication">
  <http-upgrade enabled="true" sasl-authentication-
  factory="management-saslauthentication"/>
    <socket interface="management"
    port="${jboss.management.http.port:9990}"/>
  </http-interface>
</management-interfaces>
```

Атрибуты интерфейса, указанные выше, относятся к объявлению именованного интерфейса, приведенному далее в файле «host.xml». Это объявление интерфейса будет использоваться для разрешения соответствующего сетевого интерфейса.

```
<interfaces>
  <interface name="management">
    <inet-address value="192.168.0.101"/>
  </interface>
</interfaces>
```

(Смотри «domain/configuration/host.xml»)

Пожалуйста, ознакомьтесь с главой «Конфигурация интерфейса» для получения более подробного объяснения того, как настроить сетевые интерфейсы.

Далее, по умолчанию главный контроллер домена настроен на проверку подлинности, поэтому необходимо добавить пользователя, который может использоваться подчиненным контроллером домена для подключения.

Используйте утилиту добавления пользователя, чтобы добавить нового пользователя, в этом примере я добавляю нового пользователя с именем «slave».

**Внимание:** функция добавления пользователя ДОЛЖНА быть запущена на главном контроллере домена, а не на подчиненном.

### 8.2 Конфигурация главного контроллера

Как только контроллер домена настроен правильно, вы можете приступить к работе с любым хостом, который должен присоединиться к домену. Настройка контроллера хоста состоит из трех шагов:

– логическое имя хоста (в пределах домена) должно быть различным;

– хост-контроллер должен знать IP-адрес контроллера домена.

Укажите четкое логичное имя для хоста. В следующем примере мы просто назовем его «slave»:

```
<host xmlns="urn:jboss:domain:3.0"
   name="slave">
   [...]
</host>
```

(Смотри «domain/configuration/host.xml»)

Если атрибут «name» не задан, именем хоста по умолчанию будет значение «jboss.host.name» системное свойство. Если это значение не задано, будет использоваться значение переменной среды «HOSTNAME» или «COMPUTERNAME», одна из которых будет установлена в большинстве операционных систем. Если ни то, ни другое не задано, именем будет значение «InetAddress.getLocalHost().getHostName()».

В подсистеме elytron должен быть определен контекст аутентификации, который будет содержать идентификатор хост-контроллера.

```
<subsystem xmlns="urn:WildBoss Pro:elytron:14.0" final-
providers="combined-providers" disallowed-providers="OracleUcrypto">
   <authentication-client>
    <authentication-configuration sasl-mechanism-selector="DIGEST-MD5"
                                  name="hostAuthConfig"
                                  authentication-name="slave"
                                  realm="ManagementRealm">
      <credential-reference clear-text="host_us3r_password"/>
    </authentication-configuration>
    <authentication-context name="hcAuthContext">
      <match-rule match-host="${jboss.test.host.master.address}"
                  authentication-configuration="hostAuthConfig"/>
    </authentication-context>
   </authentication-client>
```

```
....
```

Скажите ему, как найти контроллер домена, чтобы он мог зарегистрироваться в этом домене:

```
<domain-controller>
  <remote protocol="remote" host="192.168.0.101" port="9999"
  authentication-context="hcAuthContext"/>
</domain-controller>
```

Поскольку также предоставили доступ к интерфейсу управления HTTP, также могли бы использовать:

```
<domain-controller>
<remote protocol="http-remoting" host="192.168.0.101" port="9990"
username="slave" authentication-context="hcAuthContext"/>
</domain-controller>
```

(Смотри domain/configuration/host.xml)

**Внимание:** имя каждого хоста должно быть уникальным при регистрации на контроллере домена, однако имя пользователя не должно быть уникальным - использование атрибута «username» позволяет использовать одну и ту же учетную запись нескольким хостам, если это имеет смысл в вашей среде.

### 8.2.1 Игнорирование общедоступных ресурсов

WildBoss Pro 10 и более поздние версии позволяют подчиненным контроллерам хоста легко «игнорировать» части конфигурации всего домена. Что это означает и почему это полезно?

Одной из обязанностей контроллера домена является обеспечение того, чтобы все работающие контроллеры хоста имели согласованную локальную копию конфигурации всего домена (т.е. те ресурсы, адрес которых не начинается «с /host=*», т.е. те, которые сохраняются в «domain.xml». Наличие этой локальной копии позволяет пользователю выполнять следующие действия:

– попросите подчиненное устройство запустить свои уже настроенные серверы, даже если контроллер домена не запущен;

– сконфигурируйте новые серверы, используя группы серверов, отличные от тех, что запущены в данный момент, и попросите подчиненное устройство запустить их, даже если контроллер домена не запущен;

– перенастройте подчиненный сервер так, чтобы он выполнял функции контроллера домена, позволяя ему выполнять функции ведущего, если предыдущий ведущий сервер вышел из строя или был выключен.

Однако из этих трех условий только последние два требуют, чтобы подчиненное устройство сохраняло полную копию конфигурации всего домена. Первое требует, чтобы подчиненное устройство имело только ту часть конфигурации всего домена, которая имеет отношение к его текущим серверам. И первый вариант использования является наиболее распространенным. Подчиненное устройство, предназначенное только для поддержки первого варианта использования, может безопасно «игнорировать» части конфигурации всего домена. И в игнорировании некоторых ресурсов есть свои преимущества:

– если группа серверов игнорируется, а развертывания, сопоставленные с этой группой серверов, не сопоставляются с другими группами, которые не игнорируются, подчиненному устройству не нужно извлекать копию содержимого развертывания из ведущего устройства.

Это может сэкономить дисковое пространство на подчиненном устройстве, повысить скорость запуска новых хостов и сократить сетевой трафик;

– WildBoss Pro поддерживает «смешанные домены», в которых контроллер домена более поздней версии может управлять подчиненными устройствами, работающими под управлением предыдущих версий. Но эти «устаревшие» подчиненные устройства не могут понимать ресурсы конфигурации, атрибуты и операции, представленные в более новых версиях. Таким образом, любая попытка использовать более новые возможности в конфигурации всего домена потерпит неудачу, если только устаревшие подчиненные устройства не игнорируют соответствующие ресурсы. Но игнорирование ресурсов позволит устаревшим подчиненным устройствам нормально работать, управляя серверами с использованием профилей без использования новых концепций, в то время как другие хосты могут запускать серверы с профилями, которые используют преимущества новейших функций.

До версии WildBoss Pro 10 ведомое устройство можно было настроить на игнорирование некоторых ресурсов, но этот механизм был не особенно удобен для пользователя:

– ресурсы, которые должны были игнорироваться, должны были быть достаточно подробно перечислены в конфигурации каждого хоста;

– если добавлен новый ресурс, который необходимо игнорировать, то каждый хост, который необходимо игнорировать, должен быть обновлен для записи этого.

Начиная с версии WildBoss Pro 10, такая подробная настройка больше не требуется. Вместо этого в стандартных версиях «host.xml» ведомое устройство будет работать следующим образом:

– если ведомый сервер был запущен с параметром командной строки «--backup», поведение будет таким же, как и в версиях до версии 10; т.е. будут игнорироваться только ресурсы, специально настроенные для игнорирования;

– в противном случае подчиненное устройство будет «игнорировать неиспользуемые ресурсы».

Что означает «игнорирование неиспользуемых ресурсов»?

– любая группа серверов, на которую не ссылается ни один из ресурсов конфигурации сервера хоста, игнорируется;

– любой профиль, на который не ссылается не игнорируемая серверная группа, прямо или косвенно через атрибут «include» ресурса профиля, игнорируется;

– любая группа привязки сокетов, на которую напрямую не ссылается ни один из ресурсов конфигурации сервера хоста, или на которую не ссылается игнорируемая группа серверов, игнорируется;

– ресурсы расширения не будут автоматически игнорироваться, даже если ни один из профилей, не являющихся игнорируемыми, не использует расширение. Игнорирование расширения требует явной настройки. Возможно, в будущем выпуске расширения будут явно игнорироваться;

– если в конфигурацию подчиненного хоста или в конфигурацию всего домена вносится изменение, которое сокращает набор игнорируемых ресурсов, то в рамках обработки этого изменения подчиненный сервер свяжется с ведущим, чтобы удалить недостающие элементы конфигурации и интегрировать эти элементы в свою локальную копию модели управления. Примеры таких изменений включают добавление нового «serverconfig» , который ссылается на ранее игнорируемую группу серверов или группу привязки сокетов, изменение группы серверов или группы привязки сокетов, назначенных конфигурации сервера, изменение профиля или группы привязки сокетов, назначенных группе серверов, которая не игнорируется, или добавление профиля или группы привязки сокетов к набору тех, которые прямо или косвенно включены в не игнорируемый профиль или группу привязки сокетов.

Поведение по умолчанию можно изменить, либо всегда игнорировать неиспользуемые ресурсы, даже если используется «--backup», либо не игнорировать неиспользуемые ресурсы,

обновив элемент «domain-controller» в файле «host- xml» и установив атрибут «ignore-unused-configuration»:

```
<domain-controller>
  <remote authentication-context="hcAuthContext" ignore-unused-
  configuration="false
  ">
    <discovery-options>
      <static-discovery name="primary"
      protocol="${jboss.domain.master.protocol:remote}"
      host="${jboss.domain.master.address}"
      port="${jboss.domain.master.port:9999}"/>
    </discovery-options>
  </remote>
</domain-controller>
```

Поведение «игнорировать неиспользуемые ресурсы» можно использовать в сочетании с подробным описанием того, что следует игнорировать, в версии WildBoss Pro 10. Если это будет сделано, будут проигнорированы как неиспользуемые ресурсы, так и явно объявленные ресурсы. Вот пример такой конфигурации, в которой подчиненный сервер не может использовать расширение «org.example.foo», которое было установлено на контроллере домена и на некоторых подчиненных устройствах, но не на этом:

```
<domain-controller>
  <remote authentication-context="hcAuthContext" ignore-unused-
  configuration="true">
    <ignored-resources type="extension">
       <instance name="org.example.foo"/>
    </ignored-resources>
    <discovery-options>
       <static-discovery name="primary"
       protocol="${jboss.domain.master.protocol:remote}"
       host="${jboss.domain.master.address}"
       port="${jboss.domain.master.port:9999}"/>
    </discovery-options>
  </remote>
</domain-controller>
```

### 8.3 Группы серверов

Контроллер домена определяет одну или несколько групп серверов и связывает каждую из них с профилем и группой привязки сокетов, а также:

```
<server-groups>
  <server-group name="main-server-group" profile="default">
    <jvm name="default">
       <heap size="64m" max-size="512m"/>
       <permgen size="128m" max-size="128m"/>
    </jvm>
    <socket-binding-group ref="standard-sockets"/>
  </server-group>
  <server-group name="other-server-group" profile="bigger">
    <jvm name="default">
       <heap size="64m" max-size="512m"/>
    </jvm>
```

```
    <socket-binding-group ref="bigger-sockets"/>
  </server-group>
</server-groups>
```

(Смотри «domain/configuration/domain.xml»)

Контроллер домена также определяет группы привязки сокетов и профили. Группы привязки сокетов определяют используемые по умолчанию привязки сокетов:

```
<socket-binding-groups>
  <socket-binding-group name="standard-sockets" default-
  interface="public">
    <socket-binding name="http" port="8080"/>
    [...]
  </socket-binding-group>
  <socket-binding-group name="bigger-sockets" include="standard-
  sockets" defaultinterface="public">
    <socket-binding name="unique-to-bigger" port="8123"/>
  </socket-binding-group>
</socket-binding-groups>
```

(Смотри «domain/configuration/domain.xml»)

В этом примере группа «bigger-sockets» включает в себя все привязки к сокетам, определенные в группах «standardsockets», а затем определяет собственную дополнительную привязку к сокетам. Профиль — это набор подсистем, и именно эти подсистемы реализуют функциональность, которую пользователи ожидают от сервера приложений.

```
<profiles>
   <profile name="default">
     <subsystem xmlns="urn:jboss:domain:web:1.0">
       <connector name="http" scheme="http" protocol="HTTP/1.1"
       socket-binding="http"/>
       [...]
     </subsystem>
     <\!-\- The rest of the subsystems here \-->
     [...]
     </profile>
     <profile name="bigger">
     <subsystem xmlns="urn:jboss:domain:web:1.0">
       <connector name="http" scheme="http" protocol="HTTP/1.1"
       socket-binding="http"/>
       [...]
     </subsystem>
     <\!-\- The same subsystems as defined by 'default' here \-->
     [...]
     <subsystem xmlns="urn:jboss:domain:fictional-example:1.0">
       <socket-to-use name="unique-to-bigger"/>
     </subsystem>
   </profile>
</profiles>
```

(Смотри «domain/configuration/domain.xml»)

Здесь у нас есть два профиля. Более крупный профиль содержит все те же подсистемы, что и профиль по умолчанию (хотя параметры для различных подсистем могут отличаться в каждом профиле), и добавляет подсистему с вымышленным примером, которая ссылается на привязку сокета «unique-to-bigger».

### 8.4 Серверы

Хост-контроллер определяет один или несколько серверов:

```
<servers>
  <server name="server-one" group="main-server-group">
    <\!-\- server-one inherits the default socket-group declared in
    the servergroup\-->
    <jvm name="default"/>
  </server>
  <server name="server-two" group="main-server-group" auto-
  start="true">
    <socket-binding-group ref="standard-sockets" port-offset="150"/>
    <jvm name="default">
      <heap size="64m" max-size="256m"/>
    </jvm>
  </server>
  <server name="server-three" group="other-server-group" auto-
  start="false">
    <socket-binding-group ref="bigger-sockets" port-offset="250"/>
  </server>
</servers>
```

(Смотри «domain/configuration/host.xml»)

Первый (server-one) и второй (server-two) серверы связаны с основной серверной группой, что означает, что они оба запускают подсистемы, определенные профилем по умолчанию, и имеют привязки к сокетам, определенные группой привязки к сокетам «standard-sockets». Поскольку все серверы, определенные хостом, будут запущены на одном и том же физическом хосте, мы бы получили конфликты портов, если бы не использовали <socket-binding-group ref="standard-sockets" port-offset="150"/> для второго сервера. Это означает, что второй сервер будет использовать привязки сокетов, определенные стандартными сокетами, но это добавит 150 к каждому определенному номеру порта, поэтому значение, используемое для «http», будет равно 8230 для второго сервера.

Третий (server-three) сервер не будет запущен из-за его автоматического «auto-start="false"». Значение по умолчанию, если автоматический запуск не задан, равно «true», поэтому при запуске хост-контроллера будут запущены как первый сервер, так и второй сервер. Третий сервер принадлежит к группе «other-server-group», поэтому, если бы его автозапуск был изменен на «true», он бы запускался с использованием подсистем из более крупного профиля и использовал бы группу привязки сокетов «bigger-sockets».

#### 8.4.1 JVM

Хост-контроллер содержит основные определения «jvm» с аргументами:

```
<jvms>
  <jvm name="default">
    <heap size="64m" max-size="128m"/>
  </jvm>
</jvms>
```

(Смотри «domain/configuration/host.xml»)

Из предыдущих примеров мы можем видеть, что у нас также была ссылка на «jvm» на уровне группы серверов в контроллере домена. Имя «jvm» должно соответствовать одному из определений в контроллере хоста. Значения, указанные на уровне контроллера домена и хост-

контроллера, объединяются, причем хост -контроллер имеет приоритет, если в обоих местах указан один и тот же параметр.

Наконец, как видно, мы также можем переопределить «jvm» на уровне сервера. Опять же, имя виртуальной машины «jvm» должно соответствовать одному из определений в контроллере хоста. Значения объединяются с теми, которые поступают с уровня контроллера домена и хост-контроллера, на этот раз определение сервера имеет приоритет, если во всех местах указан один и тот же параметр.

В соответствии с этими правилами параметрами «jvm» для запуска каждого сервера будут следующие.

Таблица 16

| **Сервер** | **Параметры JVM** |
|---|---|
| server-one | -Xms64m -Xmx128m |
| server-two | -Xms64m -Xmx256m |
| server-three | -Xms64m -Xmx128m |

# 9 Задачи управления

## 9.1 Параметры командной строки

Чтобы запустить домен, управляемый WildBoss Pro, выполните скрипт «$JBOSS_HOME/bin/domain.sh». Чтобы запустить автономный сервер, выполните команду «$JBOSS_HOME/bin/standalone.sh». Без аргументов используется конфигурация по умолчанию. Вы можете переопределить конфигурацию по умолчанию, указав аргументы в командной строке или в вызывающем скрипте.

### 9.1.1 Свойства системы

Чтобы задать системное свойство, передайте его новое значение, используя стандартные параметры «jvm -Dkey=value»:

```
$JBOSS_HOME/bin/standalone.sh -Djboss.home.dir=some/location/WildBoss
Pro \
  -Djboss.server.config.dir=some/location/WildBoss Pro/custom-
  standalone
```

Эта команда запускает автономный экземпляр сервера, используя нестандартный домашний каталог AS и каталог пользовательской конфигурации. Для получения дополнительной информации о свойствах системы обратитесь к определениям, приведенным ниже.

Вместо того чтобы передавать параметры напрямую, вы можете поместить их в файл свойств и передать файл свойств скрипту, как в двух примерах ниже.

```
$JBOSS_HOME/bin/domain.sh --properties=/some/location/jboss.properties
$JBOSS_HOME/bin/domain.sh -P=/some/location/jboss.properties
```

Однако обратите внимание, что свойства, заданные таким образом, не обрабатываются при запуске JVM. Они обрабатываются на ранней стадии процесса загрузки, но этот механизм не следует использовать для настройки свойств, управляющих поведением JVM (например, «java.net.perferIPv4Stack») или поведением системы загрузки классов модулей JBoss.

Синтаксис для передачи файлов параметров и свойств одинаков независимо от того, используете ли «domain.sh», «standalone.sh» или Microsoft Windows scripts «domain.bat» или «standalone.bat».

Файл свойств представляет собой стандартный файл свойств Java, содержащий пары «key=value»:

```
jboss.home.dir=/some/location/WildBoss Pro
jboss.domain.config.dir=/some/location/WildBoss Pro/custom-domain
```

Системные свойства также можно задать с помощью файлов конфигурации «xml». Однако обратите внимание, что для автономного сервера свойства, заданные таким образом, не будут установлены до тех пор, пока не будет проанализирована конфигурация «xml» и не будут выполнены команды, созданные синтаксическим анализатором. Таким образом, этот механизм не следует использовать для установки свойств, значение которых должно быть задано до этого момента.

*Управление расположениями файловой системы с помощью системных свойств* - в автономном режиме и в режиме управляемого домена используется конфигурация по умолчанию, в соответствии с которой различные файлы и доступные для записи каталоги

должны находиться в стандартных расположениях. Каждое из этих стандартных расположений связано с системным свойством, которое имеет значение по умолчанию. Чтобы переопределить системное свойство, передайте его новое значение, используя один из описанных выше механизмов. С помощью системного свойства можно управлять следующими местоположениями:

Таблица 17 – Автономный

| Название объекта | Использование | Значение по умолчанию |
|---|---|---|
| java.ext.dirs | Пути к каталогам расширений JDK | null |
| jboss.home.dir | Корневой каталог установки WildBoss Pro | Устанавливается с помощью «standalone.sh» в «$JBOSS_HOME» |
| jboss.server.base.dir | Базовый каталог для содержимого сервера | jboss.home.dir/standalone |
| jboss.server.config.dir | Каталог базовой конфигурации | jboss.server.base.dir/configuration |
| jboss.server.data.dir | Каталог, используемый для постоянного хранения файлов данных | jboss.server.base.dir/data |
| jboss.server.log.dir | Каталог, содержащий файл «server.log» | jboss.server.base.dir/log |
| jboss.server.temp.dir | Каталог, используемый для временного хранения файлов | jboss.server.base.dir/tmp |
| jboss.server.deploy.dir | Каталог, используемый для хранения развернутого содержимого | jboss.server.data.dir/content |

Таблица 18 - Управляемый домен

| Название объекта | Использование | Значение по умолчанию |
|---|---|---|
| jboss.home.dir | Корневой каталог установки WildBoss Pro | Устанавливается с помощью «domain.sh» в «$JBOSS_HOME» |
| jboss.domain.base.dir | Базовый каталог для содержимого домена | jboss.home.dir/domain |
| jboss.domain.config.dir | Каталог базовой конфигурации | jboss.domain.base.dir/configuration |
| jboss.domain.data.dir | Каталог, используемый для постоянного хранения файлов данных | jboss.domain.base.dir/data |
| jboss.domain.log.dir | Каталог, содержащий host-controller.log и processcontroller.log files | jboss.domain.base.dir/log |
| jboss.domain.temp.dir | Каталог, используемый для временного хранения файлов | jboss.domain.base.dir/tmp |
| jboss.domain.deployment.dir | Каталог, используемый для хранения развернутого содержимого | jboss.domain.base.dir/content |

| Название объекта | Использование | Значение по умолчанию |
|---|---|---|
| jboss.domain.servers.dir | Каталог, содержащий выходные данные для управляемых экземпляров сервера | jboss.domain.base.dir/servers |

### 9.1.2 Другие параметры командной строки

Первым приемлемым форматом аргументов командной строки для сценариев запуска WildBoss Pro является.

```
--name=value
```

Например:

```
$JBOSS_HOME/bin/standalone.sh --server-config=standalone-ha.xml
```

Если имя параметра состоит из одного символа, то перед ним ставится один «-» вместо двух. Некоторые параметры могут быть как длинными, так и короткими.

```
-x=value
```

Например:

```
$JBOSS_HOME/bin/standalone.sh -P=/some/location/jboss.properties
```

Для некоторых аргументов командной строки, часто используемых в предыдущих основных версиях WildBoss Pro, в приведенных выше примерах поддерживается замена «=» на пробел для обеспечения совместимости.

```
-b 192.168.100.10
```

Если возможно, используйте синтаксис «-x=value». Новые параметры всегда будут поддерживать этот синтаксис.

В разделах ниже описаны имена параметров командной строки, которые доступны в автономном режиме и в режиме домена.

Таблица 19 - Автономный

| Имя | По умолчанию, если отсутствует | Значение |
|---|---|---|
| --admin-only | - | установите для сервера тип запуска «ADMIN_ONLY», чтобы он открывал административные интерфейсы и принимал запросы управления, но не |

| Имя | По умолчанию, если отсутствует | Значение |
|---|---|---|
| | | запускал другие службы времени выполнения и не принимал запросы конечных пользователей |
| --server-config -c | standalone.xml | относительный путь, который интерпретируется как относительный к «jboss.server.config.dir». Имя используемого файла конфигурации |
| --read-only-server-config | - | относительный путь, который интерпретируется как относительный к «jboss.server.config.dir». Это аналогично «--server-config», но если указана эта альтернатива, сервер не будет перезаписывать файл при изменении модели управления. Однако для файла сохраняется полная история версий |
| --graceful-startup | true | запускайте серверы корректно, ставя их в очередь или автоматически отклоняя входящие запросы, пока сервер не будет полностью запущен |
| --git-repo | - | URL-адрес удаленного репозитория Git для использования в качестве каталога конфигурации и содержимого репозитория содержимого или локального, если будет использоваться только локальный репозиторий |
| --git-branch | master | ветка Git или тег, который будет использоваться. Если используется имя тега, то будущие коммиты перейдут в состояние отсоединения |
| --git-auth | - | URL -адрес файла конфигурации Elytron, содержащего учетные данные, которые будут использоваться для подключения к репозиторию Git |

Таблица 20 - Управляемый домен

| Имя | По умолчанию, если отсутствует | Значение |
|---|---|---|
| --admin-only | - | Установите для запущенного сервера значение «ADMIN_ONLY», чтобы он открывал административные интерфейсы и принимал запросы на управление, но не запускал серверы, или, если этот контроллер хоста является главным для домена, принимал входящие соединения от подчиненных контроллеров хоста |
| --domain-config -c | domain.xml | Относительный путь, который интерпретируется как относительный к «jboss.domain.config.dir». Имя файла |

| Имя | По умолчанию, если отсутствует | Значение |
|---|---|---|
| | | конфигурации для всего домена, который будет использоваться |
| --read-only-domain-config | - | Относительный путь, который интерпретируется как относительный к «jboss.domain.config.dir». Это похоже на «--domain-config», но если указана эта альтернатива, контроллер хоста не будет перезаписывать файл при изменении модели управления. Однако для файла сохраняется полная история версий |
| --host-config | host.xml | Относительный путь, который интерпретируется как относительный к «jboss.domain.config.dir». Имя используемого файла конфигурации для конкретного хоста |
| --read-only-host-config | - | Относительный путь, который интерпретируется как относительный к «jboss.domain.config.dir». Это аналогично – «-host-config», но если указана эта альтернатива, контроллер хоста не будет перезаписывать файл при изменении модели управления. Однако для файла сохраняется полная история версий |

Следующие параметры не имеют значения и применимы только к подчиненным хост-контроллерам (т.е. хостам, настроенным для подключения к удаленному контроллеру домена).

Таблица 21

| Имя | Функция |
|---|---|
| --backup | Заставляет подчиненный host-контроллер создавать и поддерживать локальную копию («domain.cachedremote. xml») конфигурации домена. Если параметр «ignore-unused-configuration» не установлен в «host.xml», полная копия конфигурации домена будет сохранена локально, в противном случае настроенное значение «ignore-unusedconfiguration» в «host.xml» будет использоваться. (Более подробную информацию смотрите в разделе «ignore-unused-configuration») |
| --cached-dc | Если подчиненный host -контроллер не может связаться с главным контроллером домена, чтобы получить его конфигурацию при загрузке, этот параметр позволит подчиненному хост-контроллеру загрузиться и начать работу, используя предварительно кэшированную копию конфигурации домена («domain.cached-remote.xml») Если кэшированная конфигурация отсутствует, эта загрузка завершится ошибкой. Этот файл создается с помощью одного из следующих способов:<br>- ранее успешно подключенного к главному контроллеру домена с помощью «--backup» или «--cached-dc»;<br>- копирования конфигурации домена с альтернативного хоста в «domain/configuration/domain.cachedremote. xml». Недоступный главный контроллер домена будет периодически проверяться на доступность, и как только он станет доступным, подчиненный контроллер хоста снова |

| Имя | Функция |
|---|---|
| | подключится к главному контроллеру хоста и синхронизирует конфигурацию домена. В течение периода, в течение которого главный контроллер домена недоступен, подчиненный контроллер хоста не сможет вносить какие-либо изменения в конфигурацию домена. Присваивает системному свойству «jboss.bind.address» значение «значение». Более подробную информацию смотрите в разделе «Управление адресом привязки с помощью «-b»» |

*Общие параметры*

Эти параметры применяются как в автономном режиме, так и в режиме управляемого домена:

Таблица 22

| Имя | Функция |
|---|---|
| -b=<value> | Присваивает системному свойству «jboss.bind.address» значение «значение». Более подробную информацию смотрите в разделе «Управление адресом привязки с помощью «-b»» |
| -b<name>=<value> | Устанавливает системное свойство «jboss.bind.address». «имя» в «значение», где имя может изменяться. Более подробную информацию смотрите в разделе «Управление адресом привязки с помощью «-b»» |
| -u=<value> | Присваивает системному свойству «jboss.default.multicast.address» значение «значение». Видеть Управление адресом многоадресной рассылки по умолчанию с помощью «-u» для получения более подробной информации |
| --version -v -V | Выводит версию WildBoss Pro на стандартный вывод и завершает работу с JVM |
| --help-h | Выводит справочное сообщение с объяснением параметров и завершает работу с виртуальной машиной JVM |

### 9.1.3 Управление адресом привязки с помощью «-b»

WildBoss Pro привязывает сокеты к IP-адресам и интерфейсам, содержащимся в элементах «интерфейсы» в «standalone.xml», «domain.xml» и «host.xml». (Дополнительную информацию об этих элементах смотрите в разделе «Интерфейсы и привязки сокетов») Стандартные конфигурации, поставляемые с WildBoss Pro, включают две конфигурации интерфейса:

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
</interfaces>
```

В этих конфигурациях используются значения системных свойств «jboss.bind.address.management» и «jboss.bind.address», если они заданы. Если они не заданы, для каждого значения используется значение «127.0.0.1».

Как указано в разделе «Общие параметры», AS поддерживает ключи командной строки «-b» и «-b.<name>». Единственная функция этих ключей - устанавливать системные свойства «jboss.bind.address» и «jboss.bind.address.<name>» соответственно. Однако, из-за способа настройки стандартных файлов конфигурации WildBoss Pro, использование переключателей «-b» может косвенно управлять тем, как AS связывает сокеты. Если конфигурации вашего интерфейса соответствуют приведенным выше, то при использовании этой команды в качестве команды запуска все сокеты, связанные с интерфейсом с именем «public», будут привязаны к «192.168.100.10».

```
$JBOSS_HOME/bin/standalone.sh -b=192.168.100.10
```

В стандартных конфигурационных файлах открытыми интерфейсами называются интерфейсы, не связанные с управлением сервером. Открытые интерфейсы обрабатывают обычные запросы конечных пользователей.

**Внимание:** интерфейс с именем «public» по своей сути не является чем-то особенным. Он предоставляется для удобства. Вы можете называть свои интерфейсы в соответствии с вашей средой.

Чтобы привязать общедоступные интерфейсы ко всем адресам IPv4 (подстановочный адрес IPv4), используйте следующий синтаксис:

```
$JBOSS_HOME/bin/standalone.sh -b=0.0.0.0
```

Вы также можете привязать интерфейсы управления следующим образом:

```
$JBOSS_HOME/bin/standalone.sh -bmanagement=192.168.100.10
```

В стандартных конфигурационных файлах интерфейсами управления являются сокеты, связанные с управлением сервером, такие как сокет, используемый CLI, HTTP-сокет, используемый консолью администратора, и гнездо для подключения разъема JMX.

Внимание: параметр «-b» управляет только привязками интерфейса, потому что стандартные конфигурационные файлы, поставляемые с WildBoss Pro, настраивают все таким образом. Если вы измените «интерфейсы» если в вашей конфигурации больше не используются системные свойства, управляемые параметром «-b», то установка параметра «-b» в вашей команде запуска не будет иметь никакого эффекта.

Например, эта совершенно правильная настройка для «общедоступного» интерфейса приводит к тому, что параметр «-b» не оказывает никакого влияния на «общедоступный» интерфейс:

```
<interface name="public">
   <nic name="eth0"/>
</interface>
```

Ключевым моментом является то, что конфигурацию определяет содержимое файлов конфигурации. Параметры, подобные «-b», не являются переопределениями файлов конфигурации. Они лишь предоставляют более короткий синтаксис для настройки системных

свойств, на которые могут быть ссылки, а могут и не быть в файлах конфигурации. Они предоставляются для удобства, и вы можете изменить свою конфигурацию, чтобы игнорировать их.

### 9.1.4 Управление адресом многоадресной рассылки по умолчанию с помощью «- u»

WildBoss Pro может использовать многоадресную рассылку для некоторых сервисов, в частности для тех, которые связаны с кластеризацией высокой доступности. Используемые многоадресные адреса и порты настраиваются с помощью элементов «socketbinding» в «standalone.xml» и «domain.xml». (Дополнительную информацию об этих элементах см. в разделе «Привязки сокетов»). Стандартные конфигурации HA, поставляемые с WildBoss Pro, включают две конфигурации привязки сокетов, в которых используется адрес многоадресной рассылки по умолчанию:

```
<socket-binding name="jgroups-mping" port="0" multicast-address=
"${jboss.default.multicast.address:230.0.0.4}" multicast-
port="45700"/>
<socket-binding name="jgroups-udp" port="55200" multicast-address=
"${jboss.default.multicast.address:230.0.0.4}" multicast-
port="45688"/>
```

В этих конфигурациях используются значения системного свойства «jboss.default.multicast.address», если оно задано. Если оно не задано, для каждого значения используется значение «230.0.0.4». (Конфигурация может включать в себя другие привязки сокетов для служб на основе многоадресной рассылки, которые не предназначены для использования адреса многоадресной рассылки по умолчанию; например, привязка, используемая службами «mod-cluster» для связи по отдельному адресу/порту с серверами Apache httpd.)

Как указано в разделе «Общие параметры», AS поддерживает параметр командной строки «-u». Единственная функция этого параметра - установить системное свойство «jboss.default.multicast.address». Однако, из-за способа настройки стандартных файлов конфигурации AS, использование ключей «-u» может косвенно управлять тем, как AS использует многоадресную рассылку.

Если ваши конфигурации привязки сокетов совпадают с приведенными выше, использование этой команды в качестве команды запуска приведет к тому, что служба, использующая эти конфигурации сокетов, будет взаимодействовать по многоадресному адресу «230.0.1.2».

```
$JBOSS_HOME/bin/standalone.sh -u=230.0.1.2
```

**Внимание:** как и в случае с переключателем «-b», переключатель «-u» управляет только используемым адресом многоадресной рассылки, поскольку стандартные конфигурационные файлы, поставляемые с WildBoss Pro, настраивают все таким образом. Если вы измените разделы «привязка к сокету» в своей конфигурации таким образом, чтобы они больше не использовали системные свойства, управляемые параметром «-u», то установка параметра «-u» в вашей команде запуска не будет иметь никакого эффекта.

## 9.2 Приостановка, возобновление и плавное завершение работы

### 9.2.1 Основные концепции

WildBoss Pro предоставляет возможность приостанавливать и возобновлять работу серверов. Это может быть объединено с отключением, чтобы сервер мог корректно завершить обработку всех активных запросов и затем завершить работу. Когда сервер приостанавливается, он немедленно прекращает принимать новые запросы, но ожидает завершения существующих запросов. Приостановленный сервер может быть возобновлен в любой момент и немедленно начнет обрабатывать запросы. Приостановка и возобновление работы не влияют на состояние развертывания (например, если сервер приостановлен, компоненты «singleton Jakarta Enterprise Beans» не будут уничтожены). Начиная с версии WildBoss Pro 11, сервер также можно запускать в режиме ожидания, что означает, что он не будет принимать запросы до тех пор, пока не будет возобновлен. Серверы также будут приостановлены во время процесса загрузки, поэтому запросы не будут приниматься до тех пор, пока процесс запуска не будет завершен на 100%.

Приостановка/возобновление работы не влияет на операции управления; операции управления все еще могут выполняться, пока сервер приостановлен. Если вы хотите выполнить операцию управления, которая повлияет на работу сервера, вы можете приостановить работу сервера, выполнить операцию, а затем возобновить работу сервера. Это позволяет завершить выполнение всех запросов и гарантирует, что ни один из них не будет запущен во время внесения изменений в управление.

**Внимание:** если вы выполняете операцию управления, когда сервер приостановлен, и ответ на эту операцию содержит заголовки ответа «operation-requires-reload» или «operationrequires- restart», то операция не вступит в силу в полном объеме, пока не будет выполнена перезагрузка или перезапуск. Простого возобновления работы сервера будет недостаточно, чтобы изменения вступили в силу.

Когда сервер приостанавливает работу, он переходит в четыре различных состояния:

– **RUNNING -** в нормальном состоянии сервер принимает запросы и работает в обычном режиме;

– **PRE_SUSPEND -** в режиме PRE_SUSPEND сервер уведомит внешние стороны о том, что он собирается приостановить работу, например, «mod_cluster» уведомит подсистему балансировки нагрузки о том, что развертывание приостанавливается. На этом этапе запросы все еще принимаются;

– **SUSPENDING -** все новые запросы отклоняются, и сервер ожидает завершения выполнения всех активных запросов. Если во время приостановки активных запросов нет, этот этап будет пропущен;

– **SUSPENDED -** все запросы завершены, и работа сервера приостановлена.

### 9.2.2 Запуск приостановлен

Чтобы перейти в приостановленный режим при использовании автономного сервера, вам необходимо добавить в командную строку «--start -mode=suspend». Также можно указать режим запуска в операции перезагрузки, чтобы сервер перезагрузился в приостановленный режим (другие возможные значения для режима запуска - обычный и только для администратора).

В режиме домена серверы можно запустить в режиме ожидания, передав параметр «suspend= true» любой команде, которая запускает, перезапускает или перезагружает сервер (например, «startservers(suspend=true)»).

### 9.2.3 Подсистема управления запросами

WildBoss Pro представляет новую подсистему под названием «Request Controller Subsystem». Эта дополнительная подсистема отслеживает все запросы в точке их ввода,

благодаря чему механизм «graceful shutdown» узнает, когда все запросы выполнены. (Это также позволяет вам установить глобальное ограничение на общее количество выполняемых запросов).

Если эта подсистема отсутствует, приостановка/возобновление работы будет ограничено. В целом, все, что происходит на этапе предварительной приостановки, будет работать как обычно (остановка доставки сообщений, уведомление подсистемы балансировки нагрузки); однако сервер не будет ждать, пока все запросы к полной и вместо этого перейти на «Подвесной режим».

С подсистемой контроллера запросов связано небольшое снижение производительности (примерно такое же, как с включением статистики), поэтому, если вам не требуется функция приостановки/возобновления работы, эту подсистему можно удалить, чтобы получить небольшой прирост производительности.

### 9.2.4 Интеграция подсистем

Приостановка/возобновление работы — это услуга, предоставляемая платформой WildBoss Pro, которую может выбрать для интеграции любая подсистема. Некоторые подсистемы интегрируются непосредственно с контроллером приостановки, в то время как другие интегрируются через подсистему контроллера запросов.

Следующие подсистемы поддерживают плавное завершение работы. Обратите внимание, что только подсистемы, которые предоставляют внешнюю точку входа на сервер, нуждаются в поддержке плавного завершения работы. Например, «Jakarta RESTful» Подсистема веб-служб не требует поддержки приостановки/возобновления работы, поскольку все доступы к «Jakarta RESTful» Веб-службы предоставляются через веб-коннектор.

– **Undertow** - обратный поток будет ждать завершения всех запросов;

– **mod_cluster** - подсистема «mod_cluster» уведомит подсистему балансировки нагрузки о том, что сервер приостанавливает работу на этапе предварительной приостановки;

– **Jakarta Enterprise Beans** - компоненты «Jakarta Enterprise Beans» будут ожидать завершения всех удаленных запросов «Jakarta Enterprise Beans» и доставки сообщений MDB. Доставка в MDB остановлена на этапе предварительной приостановки. Таймеры «Jakarta Enterprise Beans» приостановлены, и пропущенные таймеры будут активированы при возобновлении работы сервера;

– **Batch** - пакетные задания будут остановлены на контрольной точке во время приостановки работы сервера. Они будут перезапущены с этой контрольной точки, когда сервер вернется в рабочий режим;

– **EE Concurrency** - сервер будет ожидать завершения всех активных заданий. Все задания, которые уже были поставлены в очередь, будут пропущены;

– **Transactions** - подсистема транзакций ожидает завершения всех запущенных транзакций, пока сервер находится в режиме ожидания. В течение этого времени сервер отказывается запускать какие-либо новые транзакции. Но любая транзакция в полете будет обслуживаться - например, сервер примет любой входящий удаленный вызов, который содержит контекст транзакции, уже запущенной на приостановленном сервере.

*Транзакции и Jakarta Enterprise Beans*

При работе с корпоративными компонентами Jakarta Enterprise Beans необходимо включить функцию плавного завершения работы, установив для атрибута «enable-graceful-txn-shutdown» значение «true». Например, в разделе подсистемы «ejb3» в «standalone.xml»:

```
<enable-graceful-txn-shutdown value="false"/>
```

По умолчанию для подсистемы «ejb» отключено автоматическое завершение работы. Причина этого заключается в том, что такое поведение может быть нежелательным в

кластерных средах, поскольку сервер уведомляет удаленных клиентов о том, что узел больше не доступен для удаленных вызовов, только после завершения транзакций. В течение этого короткого промежутка времени клиент кластера может отправить новый запрос на завершающий работу узел, и он отклонит запрос, поскольку он не связан с существующей транзакцией. Если для этого атрибута «enable-graceful-txn-shutdown» установлено значение «false», мы отключаем режим «graceful» и Клиенты Jakarta Enterprise Beans не будут пытаться вызвать узел при его приостановке, независимо от активных транзакций.

### 9.2.5 Автономный режим

Приостановкой/возобновлением работы можно управлять с помощью следующих операций и команд CLI в автономном режиме:

```
:suspend(suspend-timeout=x)
```

Приостанавливает работу сервера. Если указано время ожидания, сервер будет ожидать завершения всех запросов в фазе приостановки в течение указанного количества секунд. Если время ожидания не указано или значение меньше нуля, сервер будет ждать бесконечно.

```
:resume
```

Возобновляет работу ранее приостановленного сервера. Сервер должен быть в состоянии немедленно начать обслуживать запросы.

```
:read-attribute(name=suspend-state)
```

Возвращает текущее состояние приостановки работы сервера.

```
shutdown --suspend-timeout=x
```

Если в команду «shutdown» будет передан параметр «timeout», то будет выполнено плавное завершение работы. Сервер будет приостановлен и будет ожидать в состоянии «ПРИОСТАНОВКИ» до указанного количества секунд завершения всех запросов перед завершением работы. Значение тайм-аута меньше нуля означает, что он будет ждать бесконечно.

### 9.2.6 Режим домена

Режим домена выполняет те же операции, что и автономный режим, однако они могут применяться на глобальном уровне, уровне группы серверов, уровне сервера и хоста:

*Весь домен*

```
:suspend-servers(suspend-timeout=x)
:resume-servers
:stop-servers(suspend-timeout=x)
```

*Группа серверов*

```
/server-group=main-server-group:suspend-servers(suspend-timeout=x)
/server-group=main-server-group:resume-servers
/server-group=main-server-group:stop-servers(suspend-timeout=x)
```

*Сервер*

```
/host=master/server-config=server-one:suspend(suspend-timeout=x)
/host=master/server-config=server-one:resume
/host=master/server-config=server-one:stop(suspend-timeout=x)
```

*Уровень хоста*

```
/host=master:suspend-servers(suspend-timeout=x)
/host=master:resume-servers
/host=master:shutdown(suspend-timeout=x)
```

Обратите внимание, что даже если сам хост-контроллер выключен, атрибут «suspend-timeout» для операции выключения на уровне хоста применяется только к серверам, а не к самому хост-контроллеру.

### 9.2.7 Плавное завершение работы по сигналу операционной системы

Если вы используете термин, подобный сигналу операционной системы, для завершения работы вашего автономного серверного процесса WildBoss Pro, например, с помощью «kill -15 <pid>», сервер WildBoss Pro автоматически завершит работу. По

умолчанию поведение будет аналогично команде CLI «shutdown --suspend-timeout=0»; то есть процесс не будет ожидать в состоянии приостановки выполнения запросов в процессе выполнения, прежде чем перейти в состояние приостановки и затем завершит работу. Другой тайм-аут можно настроить, установив системное свойство «org.WildBoss Pro.sigterm.suspend.timeout». Значение свойства должно быть целым числом, указывающим максимальное количество секунд ожидания выполнения запросов в полете. Значение «-1» означает, что сервер должен ждать бесконечно.

Плавное завершение работы по сигналу операционной системы не будет работать, если серверная JVM настроена на отключение обработки сигналов (т.е. с аргументом «-Xrs» для «java»). Это также не сработает, если метод, используемый для завершения процесса, не приведет к появлению сигнала, на который JVM сможет отреагировать (например, «kill -9»).

В управляемом домене процессы Process Controller и Host Controller не будут пытаться выполнить какое-либо корректное завершение работы в ответ на сигнал. Сервер в режиме домена может, но надлежащим способом управления жизненным циклом серверного процесса в режиме домена является использование API управления и его управляющих функций. Хост-контроллер, а не через прямые сигналы серверному процессу.

### 9.2.8 Некорректный запуск

По умолчанию WildBoss Pro запускается автоматически, что означает, что входящие запросы ставятся в очередь или полностью отклоняются до тех пор, пока сервер не будет готов к их обработке. Однако в некоторых случаях может оказаться желательным разрешить серверу начать обработку запросов как можно раньше. Одним из таких примеров может быть ситуация, когда двум развернутым приложениям необходимо взаимодействовать друг с другом во время развертывания или запуска приложения. В одном из таких сценариев приложению A необходимо отправить запрос «REST» к приложению B, чтобы получить информацию, необходимую для его собственного запуска. При плавном запуске запрос к приложению B будет заблокировано до тех пор, пока сервер не будет запущен полностью. Однако сервер не может запуститься полностью, поскольку приложение A ожидает получения данных от приложения B до завершения его развертывания/запуска. В этой ситуации возникает взаимоблокировка, и время запуска сервера истекает.

Некорректный запуск предназначен для устранения этой ситуации, поскольку он позволит WildBoss Pro как можно скорее начать пытаться отвечать на запросы. В приведенном выше сценарии, предполагающем, что приложение B успешно развернуто/запущено, приложение A также может быть запущено немедленно, поскольку его запрос будет выполнен. Однако обратите внимание, что может возникнуть ситуация «гонки»: если приложение B еще не развернуто (например, неправильный порядок развертывания или B не завершил запуск), то приложение A все равно может не запуститься, поскольку приложение B недоступно. Пользователи WildBoss Pro, использующие некорректный запуск, должны знать об этом и принимать меры для устранения подобных ситуаций. Однако при некорректном запуске WildBoss Pro больше не будет причиной сбоя развертывания в такой конфигурации.

Здесь важно обсудить, как это связано с перезагрузкой и перезапуском, а также с приостановленными запусками. При перезагрузке служба «ApplicationServerService» останавливается и запускается новая. Это эквивалентно тому, как если бы он запускался в первый раз: происходит все то же самое, но быстрее, потому что большая загрузка классов и статическая инициализация не должны повторяться. Это включает в себя соблюдение принципа плавного запуска, поэтому, если сервер изначально был запущен некорректно, он будет перезагружен таким же образом

Перезапуск сервера аналогичен. Перезапуск означает создание новой виртуальной машины JVM, поэтому вся инициализация выполняется заново точно так же, как это было при первом запуске. При перезапуске в режиме домена хост-контроллер просто перечитывает конфигурационный файл и выполняет те же действия, что и в первый раз. В режиме «standalone» перезапуск выполняется с помощью «standalone.[sh|ps1|bat]». Запущенная JVM

завершает работу с определенным кодом завершения, который скрипт распознает и запускает новый сервер, используя те же параметры, что и при первом запуске, поэтому, если вы запустите сервер некорректно, вы перезапустите сервер некорректно.

Наконец, есть «start-mode=suspend». В случае, если администратор указывает приостановленный запуск, а также некорректный запуск, приостановленный запуск будет «выигрышным». Другими словами, сервер запустится в приостановленном режиме, параметр «graceful-start=false» будет проигнорирован, и сервер зарегистрирует сообщение, указывающее на то, что это происходит.

### 9.3 Запуск и остановка серверов в управляемом домене

Запуск автономного сервера осуществляется с помощью скрипта «bin/standalone.sh». Однако в управляемом домене экземпляры сервера управляются контроллером домена и должны запускаться на уровне управления:

Прежде всего, узнайте, какие серверы настроены на конкретном хосте:

```
[domain@localhost:9990 /] :read-children-names(child-type=host)
{
 "outcome" => "success",
 "result" => ["local"]
}
[domain@localhost:9990 /] /host=local:read-children-names(child-
type=server-config)
{
 "outcome" => "success",
 "result" => [
   "my-server",
   "server-one",
   "server-three"
 ]
}
```

Теперь, когда мы знаем, что на хосте «local» настроены два сервера, мы можем продолжить и проверить их статус:

```
[domain@localhost:9990 /] /host=local/server-config=server-one:read-
resource(includeruntime=true)
{
 "outcome" => "success",
 "result" => {
   "auto-start" => true,
   "group" => "main-server-group",
   "interface" => undefined,
   "name" => "server-one",
   "path" => undefined,
   "socket-binding-group" => undefined,
   "socket-binding-port-offset" => undefined,
   "status" => "STARTED",
   "system-property" => undefined,
   "jvm" => {"default" => undefined}
 }
}
```

Вы можете изменить состояние сервера с помощью операций «start» и «stop»

```
[domain@localhost:9990 /] /host=local/server-config=server-one:stop
{
 "outcome" => "success",
 "result" => "STOPPING"
}
```

**Внимание:** навигация по топологии домена намного упрощается, когда вы используете веб-интерфейс.

### 9.4 Настройки JVM

Настройка параметров JVM отличается для управляемого домена и автономного сервера. В управляемом домене компоненты контроллера домена отвечают за запуск и остановку серверных процессов и, следовательно, определяют настройки JVM. Что касается автономного сервера, то за это отвечает процесс, запустивший сервер (например, передающий их в качестве аргументов командной строки).

#### 9.4.1 Управляемый домен

В управляемом домене настройки JVM могут быть объявлены в разных областях: для определенной группы серверов, для хоста или для конкретного сервера. Если настройки не объявлены, они наследуются из родительской области. Это позволяет настраивать или расширять параметры JVM на каждом уровне.

Давайте взглянем на объявление JVM для группы серверов:

```
<server-groups>
  <server-group name="main-server-group" profile="default">
    <jvm name="default">
       <heap size="64m" max-size="512m"/>
    </jvm>
    <socket-binding-group ref="standard-sockets"/>
  </server-group>
  <server-group name="other-server-group" profile="default">
    <jvm name="default">
       <heap size="64m" max-size="512m"/>
    </jvm>
    <socket-binding-group ref="standard-sockets"/>
  </server-group>
</server-groups>
```

(Смотри «domain/configuration/domain.xml»)

В этом примере группа серверов «main-server-group» объявляет размер кучи в 64 млн и максимальный размер кучи в 512 млн. Любой сервер, принадлежащий к этой группе, унаследует эти настройки. Вы можете изменить эти настройки для группы в целом или для конкретного сервера или хоста:

```
<servers>
 <server name="server-one" group="main-server-group" auto-
 start="true">
   <jvm name="default"/>
 </server>
 <server name="server-two" group="main-server-group" auto-
 start="true">
   <jvm name="default">
     <heap size="64m" max-size="256m"/>
   </jvm>
```

```
    <socket-binding-group ref="standard-sockets" port-offset="150"/>
  </server>
  <server name="server-three" group="other-server-group" auto-
  start="false">
    <socket-binding-group ref="standard-sockets" port-offset="250"/>
  </server>
</servers>
```

(Смотри «domain/configuration/host.xml»)

В этом случае второй сервер принадлежит к группе основных серверов и наследует настройки JVM с именем «default», но объявляет меньший максимальный размер кучи.

```
[domain@localhost:9999 /] /host=local/server-config=server-
two/jvm=default:readresource
{
   "outcome" => "success",
   "result" => {
     "heap-size" => "64m",
     "max-heap-size" => "256m",
   }
}
```

Использование расположений файловой системы в качестве параметров JVM в режиме домена.

В разделе «Управление расположениями файловой системы с помощью системных свойств» описаны доступные системные свойства, связанные с соответствующими путями к файловой системе WildBoss Pro. В дополнение ко всем свойствам режима домена, следующие свойства, относящиеся к серверу, также доступны для разрешения в качестве параметров JVM:

– jboss.server.base.dir;

– jboss.server.log.dir;

– jboss.server.data.dir;

– jboss.server.temp.dir.

Эта возможность полезна, когда вам нужно настроить параметры JVM без указания конкретного имени сервера. Например, если вы хотите перенаправить ведение журнала GC в файл в каталог сервера журналов по умолчанию, вы можете настроить следующий параметр JVM на уровне хоста:

```
[domain@localhost:9990 /] /host=master/jvm=default:add-jvm-option(jvm-
option="-Xlog:gc:file=${jboss.server.log.dir}/gc.log")
{
   "outcome" => "success",
   "result" => undefined,
     "server-groups" => {"main-server-group" => {"host" => {"master"
     => {"server-two"
     => {"response" => {
     "outcome" => "success",
     "response-headers" => {
     "operation-requires-restart" => true,
     "process-state" => "restart-required"
     }
   }}}}}}
}
```

### 9.4.2 Автономный сервер

Для автономного сервера вы должны передать настройки JVM либо в качестве аргументов командной строки при выполнении скрипта «$JBOSS_HOME/bin/standalone.sh», либо объявив их в «$JBOSS_HOME/bin/standalone.conf» (Для пользователей Windows скрипт для выполнения называется «%JBOSS_HOME%/bin/standalone.bat», в то время как настройки JVM могут быть объявлены в «%JBOSS_HOME%/bin/standalone.conf.bat»).

## 9.5 Ведение журнала аудита

В WildBoss Pro встроено ведение журнала аудита для операций управления, влияющих на модель управления. По умолчанию это отключено. Информация выводится в виде записей JSON.

Конфигурация ведения журнала аудита по умолчанию «standalone.xml» выглядит следующим образом:

```
   <management>
     <security-realms>
...
     </security-realms>
     <audit-log>
       <formatters>
         <json-formatter name="json-formatter"/>
       </formatters>
       <handlers>
         <file-handler name="file" formatter="json-formatter"
         path="auditlog.log" relative-to="jboss.server.data.dir"/>
       </handlers>
       <logger log-boot="true" log-read-only="true" enabled="false">
         <handlers>
           <handler name="file"/>
         </handlers>
       </logger>
     </audit-log>
...
```

Глядя на это через интерфейс командной строки, получается, что

```
[standalone@localhost:9990 /] /core-
service=management/access=audit:readresource(recursive=true)
{
 "outcome" => "success",
 "result" => {
   "file-handler" => {"file" => {
     "formatter" => "json-formatter",
     "max-failure-count" => 10,
     "path" => "audit-log.log",
     "relative-to" => "jboss.server.data.dir"
   }},
   "json-formatter" => {"json-formatter" => {
     "compact" => false,
     "date-format" => "yyyy-MM-dd HH:mm:ss",
     "date-separator" => " - ",
     "escape-control-characters" => false,
     "escape-new-line" => false,
     "include-date" => true
   }},
```

```
      "logger" => {"audit-log" => {
      "enabled" => false,
      "log-boot" => true,
      "log-read-only" => false,
      "handler" => {"file" => {}}
    }},
    "syslog-handler" => undefined
  }
}
```

Чтобы включить его через CLI, вам нужно всего лишь

```
[standalone@localhost:9990 /] /core-
service=management/access=audit/logger=auditlog:
write-attribute(name=enabled,value=true)
{"outcome" => "success"}
```

Данные аудита хранятся в отдельном «standalone/data/audit-log.log».

**Внимание:** подсистема ведения журнала аудита имеет множество внутренних зависимостей, и она регистрирует операции, изменяющие, включающие и отключающие ее компоненты. При настройке или изменении чего-либо во время выполнения рекомендуется вносить эти изменения как часть пакета командной строки. Например, если вы добавляете обработчик системного журнала, вам нужно добавить обработчик и его информацию за один шаг. Аналогично, если вы используете обработчик файла и хотите изменить его путь и относительные атрибуты, это должно произойти за один шаг.

### 9.5.1 Средство форматирования JSON

Первое, что необходимо настроить, — это форматировщик, в настоящее время мы поддерживаем вывод записей журнала в формате JSON. Вы можете определить несколько форматировщиков для использования с разными обработчиками. Запись журнала имеет следующий формат, и задача форматировщика - отформатировать представленные данные:

```
2013-08-12 11:01:12 - {
  "type" : "core",
  "r/o" : false,
  "booting" : false,
  "version" : "8.0.0.Alpha4",
  "user" : "$local",
  "domainUUID" : null,
  "access" : "NATIVE",
  "remote-address" : "127.0.0.1/127.0.0.1",
  "success" : true,
  "ops" : [JMX|WFLY8:JMX subsystem configuration],
    "operation" : "write-attribute",
    "name" : "enabled",
    "value" : true,
    "operation-headers" : {"caller-type" : "user"}
  }]
}
```

Он включает в себя необязательную временную метку, а затем следующую информацию в записи «json».

Таблица 23

| Имя поля | Описание |
|---|---|
| type | это может иметь значение «core», означающее, что это операция управления, или «jmx», означающее, что это происходит из подсистемы «jmx» (смотрите подсистему «jmx» для настройки журнала аудита подсистемы «jmx») |
| r/o | значение «true», если операция не изменяет модель управления, в противном случае значение «false» |
| booting | «true», если операция была выполнена во время процесса загрузки, «false», если она была выполнена после запуска сервера |
| version | номер версии экземпляра WildBoss Pro |
| user | имя пользователя, прошедшего проверку подлинности. В этом случае операция была зарегистрирована через интерфейс командной строки на том же компьютере, что и запущенный сервер, поэтому используется специальный параметр «$local user» |
| domainUUID | идентификатор, позволяющий связать воедино все операции, передаваемые с контроллера Doman на ID -серверы, контроллеры подчиненных хостов и подчиненный хост-сервер Серверы-контроллеры |
| access | это может иметь одно из следующих значений: «*NATIVE» - операция была выполнена через собственный интерфейс управления, например «CLI*HTTP» - операция была выполнена через HTTP-интерфейс домена, например, консоль администратора «*JMX» - операция была выполнена через подсистему JMX. Как настроить ведение журнала аудита для JMX, смотрите в разделе JMX |
| remote-address | адрес клиента, выполняющего эту операцию |
| success | «true», если операция была выполнена успешно, «false», если она была отменена |
| ops | выполняемые операции. Это список операций, преобразованных в формат JSON. При загрузке это будут все операции, полученные в результате синтаксического анализа «xml». После загрузки список обычно будет содержать только одну запись |

Ресурс «json formatter» имеет следующие атрибуты:

Таблица 24

| Имя поля | Описание |
|---|---|
| include-date | логическое значение, указывающее, включать или не включать временную метку в отформатированные записи журнала |
| date-separator | строка, содержащая символы, разделяющие дату и остальную часть отформатированного сообщения журнала. Будет проигнорировано, если «include-date=false» |
| date-format | формат даты, используемый для временной метки в соответствии с «java.text.SimpleDateFormat». Будет проигнорирован, если «include-date=false» |
| compact | если значение «true», то JSON будет отформатирован в одну строку. Все еще могут быть значения, содержащие новые строки, поэтому, если важно разместить всю запись в одной строке, установите значение «escape-new-line» или «escapecontrol- characters» равным «true» |

| Имя поля | Описание |
|---|---|
| escape-control-characters | если значение «true», то все управляющие символы (записи «ascii» с десятичным значением < 32) будут заменены кодом «ascii» в восьмеричном формате, например, новая строка станет «'#012'». Если это верно, то это переопределит «escape-newline= false» |
| escape-new-line | если значение «true», то все новые строки будут экранированы кодом «ascii» в восьмеричном формате, например «#012» |

### 9.5.2 Обработчики

Обработчик отвечает за сбор отформатированных данных и их регистрацию в определенном месте. В настоящее время существует два типа обработчиков: файловый и системный журнал. Вы можете настроить несколько обработчиков каждого типа и использовать их для регистрации информации.

*Обработчик файлов.*

Обработчики файлов записывают записи журнала аудита в файл на сервере. Атрибутами обработчика файлов являются:

Таблица 25

| Атрибут | Описание | Только для чтения |
|---|---|---|
| formatter | имя средства форматирования в формате JSON, используемого для форматирования записей журнала | false |
| path | путь к файлу журнала аудита | false |
| relative-to | имя другого ранее названного пути или одного из стандартных путей, предоставляемых системой. Если задан параметр "относительно", значение атрибута path рассматривается как относительное к пути, указанному этим атрибутом | false |
| failure-count | количество сбоев ведения журнала с момента инициализации обработчика | true |
| max-failure-count | Максимальное количество ошибок ведения журнала до отключения этого обработчика | false |
| disabled-due-to-failure | значение «true», если этот обработчик был отключен из-за сбоев ведения журнала | true |

В нашей стандартной конфигурации «path=audit-log.log» и «relative-to=jboss.server.data.dir», как правило, это будет «$JBOSS_HOME/standalone/data/audit-log.log».

*Обработчик системного журнала* - в конфигурации по умолчанию ведение журнала аудита системного журнала не настроено. Системный журнал - лучший выбор для ведения журнала аудита, поскольку вы можете подключиться к удаленному серверу системного журнала и обеспечить аутентификацию по протоколу TLS с помощью проверки подлинности по сертификату клиента. Серверы системного журнала сильно различаются по своим возможностям, поэтому не все настройки в этом разделе применимы ко всем серверам системного журнала. Мы тестировали с помощью «rsyslog».

Адрес обработчика системного журнала – «/core-service=management/access=audit/syslog-handler=*», и , как и в случае с обработчиками файлов,

вы можете добавлять столько записей системного журнала, сколько захотите. Ресурсы обработчика системного журнала довольно подробно ссылаются на основные RFC для системного журнала, для справки их можно найти по адресу:

– http://www.ietf.org/rfc/rfc3164.txt;
– http://www.ietf.org/rfc/rfc5424.txt;
– http://www.ietf.org/rfc/rfc6587.txt.

Ресурс обработчика системного журнала имеет следующие атрибуты:

Таблица 26

| **Форматировщик** | **Имя средства форматирования в формате JSON, используемого для форматирования записей журнала** | **false** |
|---|---|---|
| failure-count | количество сбоев ведения журнала с момента инициализации обработчика | true |
| max-failure-count | максимальное количество ошибок ведения журнала до отключения этого обработчика | false |
| disabled-due-to-failure | значение «true», если этот обработчик был отключен из-за сбоев ведения журнала | true |
| syslog-format | следует ли устанавливать формат системного журнала, указанный в RFC-5424 или RFC-3164 | false |
| max-length | максимально допустимая длина сообщения журнала, включая заголовок, в байтах. Если значение не определено, то по умолчанию оно будет равно 1024 байтам, если формат системного журнала равен RFC3164, или 2048 байтам, если формат системного журнала равен RFC5424 | false |
| truncate | Следует ли усекать сообщение, включая заголовок, если его длина в байтах превышает максимальную длину. Если задано значение «false», сообщения будут разделяться и отправляться с одинаковыми значениями заголовка | false |

При добавлении обработчика системного журнала вам также необходимо указать протокол, который он будет использовать для связи с сервером системного журнала. Допустимыми вариантами протокола являются UDP, TCP и TLS. Протокол должен быть добавлен одновременно с добавлением обработчика системного журнала, иначе произойдет сбой. Кроме того, вы можете добавить только один протокол для обработчика.

**UDP -** настраивает обработчик на использование протокола UDP для связи с сервером системного журнала. Адрес сервера ресурса UDP - «/core-service=management/access=audit/syslog-handler=*/protocol=udp». Атрибутами ресурса UDP являются:

Таблица 27

| **Атрибут** | **Описание** |
|---|---|
| host | хост сервера системного журнала для tcp-запросов |
| port | порт сервера системного журнала, принимающего tcp -запросы |
| message-transfer | настройте передачу сообщений, как описано в разделе 3.4 документа RFC-6587. Это может быть либо «OCTET_COUNTING», как описано в разделе 3.4.1 RFC-6587, либо «NON_TRANSPARENT_FRAMING», как описано в разделе 3.4.1 RFC-6587 |

**TLS** - Настраивает обработчик на использование TLC для безопасного взаимодействия с сервером системного журнала. Адрес сервера системного журнала ресурса TLS – «/core-service=management/access=audit/syslog-handler=*/protocol=tls». Атрибуты ресурса TLS такие же, как и для TCP:

Таблица 28

| Атрибут | Описание |
|---|---|
| host | хост сервера системного журнала для tls-запросов |
| port | порт сервера системного журнала, принимающего tls -запросы |
| message-transfer | настройте передачу сообщений, как описано в разделе 3.4 документа RFC-6587. Это может быть либо «OCTET_COUNTING», как описано в разделе 3.4.1 RFC-6587, либо «NON_TRANSPARENT_FRAMING», как описано в разделе 3.4.1 RFC-6587 |

TLS с аутентификацией по клиентскому сертификату.

Если вы настроили сервер системного журнала так, чтобы он требовал проверки подлинности по сертификату клиента, при создании обработчика вам также потребуется настроить хранилище сертификатов клиента, содержащее сертификат, который будет передан серверу системного журнала. Адрес ресурса хранилища клиентских сертификатов «/coreservice=management/access=audit/syslog-handler=*/protocol=tls/authentication= clientcertificate-store» и его атрибутами являются:

Таблица 29

| Атрибут | Описание |
|---|---|
| keystore-password | пароль для хранилища ключей |
| key-password | пароль для ключа хранилища ключей |
| keystore-path | путь к хранилищу ключей |
| keystore-relative-to | имя другого ранее указанного пути или одного из стандартных путей, предоставляемых системой. Если указано значение «keystore-relative-to», значение атрибута «keystore-path» обрабатывается как относительное к пути, указанному этим атрибутом |

### 9.5.3 Конфигурация регистратора

Последней частью, требующей настройки, является регистратор операций управления. Он ссылается на один или несколько обработчиков и настраивается по адресу «/core-service=management/access=audit/logger=auditlog». Атрибутами этого ресурса являются:

Таблица 30

| Атрибут | Описание |
|---|---|
| enabled | значение «true», чтобы включить ведение журнала операций управления |
| log-boot | значение «true» для регистрации операций управления при загрузке сервера, значение «false» в противном случае |
| log-read-only | если значение «true», то все операции будут записываться в журнал аудита, если значение «false», то будут записываться только операции, которые изменяют модель |

Затем, обработчики, которые используются для регистрации операций управления, настраиваются как «handler=*» дочерние элементы регистратора.

### 9.5.4 Режим домена (конфигурация для конкретного хоста)

В режиме домена ведение журнала аудита настраивается для каждого хоста в отдельном файле host.xml. Это означает, что при подключении к контроллеру домена конфигурация ведения журнала аудита отображается в записи хоста, например, здесь приведена конфигурация по умолчанию:

```
[domain@localhost:9990 /] /host=master/core-
service=management/access=audit:readresource(recursive=true)
{
   "outcome" => "success",
   "result" => {
     "file-handler" => {
       "host-file" => {
         "formatter" => "json-formatter",
         "max-failure-count" => 10,
         "path" => "audit-log.log",
         "relative-to" => "jboss.domain.data.dir"
       },
       "server-file" => {
         "formatter" => "json-formatter",
         "max-failure-count" => 10,
         "path" => "audit-log.log",
         "relative-to" => "jboss.server.data.dir"
       }
     },
     "json-formatter" => {"json-formatter" => {
       "compact" => false,
       "date-format" => "yyyy-MM-dd HH:mm:ss",
       "date-separator" => " - ",
       "escape-control-characters" => false,
       "escape-new-line" => false,
       "include-date" => true
     }},
     "logger" => {"audit-log" => {
       "enabled" => false,
       "log-boot" => true,
       "log-read-only" => false,
       "handler" => {"host-file" => {}}
     }},
     "server-logger" => {"audit-log" => {
       "enabled" => false,
       "log-boot" => true,
       "log-read-only" => false,
       "handler" => {"server-file" => {}}
     }},
     "syslog-handler" => undefined
   }
}
```

Теперь у нас есть два обработчика файлов: один называется host-файл, используемый для настройки файла для ведения журнала операций управления на хосте, и другой называется server-файл, используемый для ведения журнала операций управления, выполняемых на

серверах. Затем «logger=audit-log» используется для настройки регистратора для контроллера хоста, ссылаясь на обработчик файла хоста. «Server-logger=audit-log» используется для настройки регистратора для управляемых серверов, ссылаясь на обработчик файла сервера. Атрибуты для «server-logger=audit-log» такие же, как и для «server-logger=audit-log» в предыдущем разделе. Независимая настройка контроллера хоста и регистраторов сервера означает, что мы можем независимо управлять ведением журнала аудита для управляемых серверов и контроллера хоста.

## 9.6 Отмена операций управления

WildBoss Pro включает в себя возможность использовать интерфейс командной строки для отмены запросов на управление, которые не выполняются в обычном режиме.

### 9.6.1 Операция отмены незавершенной операции

Операция «Отмена незавершенной операции» предписывает целевому процессу найти любую операцию, которая не выполняется нормально, и отменить ее.

На отдельном сервере:

```
[standalone@localhost:9990 /] /core-
service=management/service=management-operations:cancel-non-
progressing-operation
{
   "outcome" => "success",
   "result" => "-1155777943"
}
```

Результирующее значение представляет собой внутренний идентификационный номер для операции, которая была отменена.

На хост-контроллере управляемого домена эквивалентный ресурс находится в разделе «host=<hostname>» дерева ресурсов управления:

```
[domain@localhost:9990 /] /host=host-a/core-
service=management/service=managementoperations:cancel-non-
progressing-operation
{
   "outcome" => "success",
   "result" => "2156877946"
}
```

Операция также может быть отменена на отдельном управляемом доменном сервере:

```
[domain@localhost:9990 /] /host=host-a/server=server-
one/coreservice=management/service=management-operations:cancel-non-
progressing-operation
{
   "outcome" => "success",
   "result" => "6497786512"
}
```

Считается, что операция выполняется неправильно, если она выполнялась с сохранением блокировки исключительной операции более 15 секунд. Операции, доступные только для чтения, не получают блокировки исключительной операции, поэтому эта операция не отменяет операции, доступные только для чтения. Блокировка операций в ожидании другой операции для снятия исключительной блокировки также не будет отменена.

Если нет ни одной операции, которая не выполнялась бы нормально, то будет получен ответ об ошибке:

```
[standalone@localhost:9990 /] /core-
service=management/service=managementoperations:cancel-non-
progressing-operation
{
   "outcome" => "failed",
   "failure-description" => "WFLYDM0089: No operation was found that
   has been holding
   the operation execution write lock for long than [15] seconds",
   "rolled-back" => true
}
```

### 9.6.2 Операция «Найти незавершенную операцию»

Чтобы просто узнать идентификатор операции, которая не выполняется в обычном режиме, но не отменять ее, используйте операцию «find-non-progressing-operation» (найти незавершенную операцию):

```
[standalone@localhost:9990 /] /core-
service=management/service=managementoperations:find-non-progressing-
operation
{
   "outcome" => "success",
   "result" => "-1155777943"
}
```

Если нет ни одной незавершенной операции, результатом по-прежнему будет успех, но результат будет неопределенным.

Как только будет известен идентификатор операции, можно будет просмотреть ресурс управления для этой операции, чтобы узнать больше о ее статусе.

### 9.6.3 Проверка состояния активной операции

Для любой выполняемой в данный момент операции существует ресурс управления, который можно запросить:

```
[standalone@localhost:9990 /] /core-
service=management/service=managementoperations/active-operation=-
1155777943:read-resource(include-runtime=true)
{
  "outcome" => "success",
  "result" => {
    "access-mechanism" => "undefined",
    "address" => [
    ("deployment" => "example")
    ],
    "caller-thread" => "management-handler-thread - 24",
    "cancelled" => false,
    "exclusive-running-time" => 101918273645L,
    "execution-status" => "awaiting-stability",
    "operation" => "deploy",
    "running-time" => 101918279999L
  }
}
```

Ответ включает в себя следующие атрибуты:

Таблица 31

| Поле | Значение |
|---|---|
| access-mechanism | механизм, используемый для отправки запроса на сервер. NATIVE, JMX, HTTP |
| address | адрес ресурса, на который нацелена операция. Значение в последнем элементе адреса будет «<hidden>», если вызывающий абонент не авторизован для обращения к целевому ресурсу операции |
| caller-thread | имя потока, который выполняет операцию |
| cancelled | была ли отменена операция |
| exclusive-status | количество времени в наносекундах, в течение которого выполнялась операция с сохраненной исключительной блокировкой выполнения операции, или «-1», если операция не имеет исключительной блокировки выполнения |
| execution-status | текущая активность операции. Подробности смотрите ниже |
| operation | название операции или «<hidden>», если вызывающий абонент не авторизован для обращения к целевому ресурсу операции |
| running-time | количество времени, в течение которого выполнялась операция, в наносекундах |

Ниже приведены значения для атрибута «exclusive-running-time» (исключительное время выполнения):

Таблица 32

| Поле | Значение |
|---|---|
| executing | вызывающий поток активно выполняется |
| awaiting-other-operation | вызывающий поток блокируется, ожидая, пока другая операция снимет блокировку исключительного выполнения |
| awaiting-stability | вызывающий поток внес изменения в служебный контейнер и ожидает стабилизации работы служебного контейнера |
| completing | операция зафиксирована и завершает выполнение |
| rolling-back | операция сворачивается |

Все выполняемые в данный момент операции можно просмотреть в одном запросе, используя операцию «read-children-resources» (чтение дочерних элементов-ресурсов):

```
[standalone@localhost:9990 /] /core-
service=management/service=managementoperations:read-children-
resources(child-type=active-operation)
{
 "outcome" => "success",
 "result" => {"-1155777943" => {
   "access-mechanism" => "undefined",
   "address" => [
   ("deployment" => "example")
   ],
   "caller-thread" => "management-handler-thread - 24",
   "cancelled" => false,
   "exclusive-running-time" => 101918273645L,
   "execution-status" => "awaiting-stability",
```

```
    "operation" => "deploy",
    "running-time" => 101918279999L
  },
  {"-1246693202" => {
    "access-mechanism" => "undefined",
    "address" => [
    ("core-service" => "management"),
    ("service" => "management-operations")
    ],
    "caller-thread" => "management-handler-thread - 30",
    "cancelled" => false,
    "exclusive-running-time" => -1L,
    "execution-status" => "executing",
    "operation" => "read-children-resources",
    "running-time" => 3356000L
  }}
}
```

### 9.6.4 Отмена определенной операции

Операция «cancel-non-progressing-operation» — это удобная операция для идентификации и отмены операции. Однако администратор может просмотреть ресурсы активных операций, чтобы идентифицировать любую операцию, а затем напрямую отменить ее, вызвав операцию отмены на ресурсе для выполнения требуемой операции.

```
[standalone@localhost:9990 /] /core-
service=management/service=managementoperations/active-operation=-
1155777943:cancel
{
   "outcome" => "success",
   "result" => undefined
}
```

### 9.6.5 Контроль времени блокировки операции

По мере выполнения операции исполняющий поток может блокироваться в различных точках, особенно во время ожидания стабилизации контейнера службы после любых изменений. Поскольку операция может удерживать исключительную блокировку выполнения во время блокировки, в WildBoss Pro поведение выполнения было изменено, чтобы гарантировать, что время ожидания блокировки в конечном итоге истечет, что приведет к откату операции.

Время ожидания блокировки по умолчанию составляет 300 секунд. Это намеренно большое время, поскольку идея заключается в том, чтобы запускать тайм-аут только в том случае, если что-то определенно пошло не так с операцией, без каких-либо ложных срабатываний.

Администратор может управлять временем ожидания блокировки для отдельной операции, используя заголовок операции «blocking-timeout». Например, если известно, что для развертывания конкретного развертывания требуется очень много времени, время ожидания по умолчанию в 300 секунд может быть увеличено:

```
[standalone@localhost:9990 /] deploy /tmp/mega.war --
headers={blocking-timeout=450}
```

Обратите внимание, что тайм-аут блокировки не является гарантированным максимальным временем выполнения операции. Если это всего лишь тайм-аут, который будет применяться в различные моменты выполнения операции.

## 9.7 История конфигурационных файлов

В процессе управления модель может быть изменена. Когда это происходит, xml-файл, поддерживающий модель, записывается заново, отражая последние изменения. Кроме того, сохраняется полная история файла. История файла хранится в отдельном каталоге в каталоге конфигурации.

Как уже упоминалось в разделе «Параметры командной строки», файл конфигурации по умолчанию можно выбрать с помощью параметра командной строки. Для автономного экземпляра сервера история активных «standalone.xml» хранится в «jboss.server.config.dir/standalone_xml_history» (более подробную информацию смотрите в разделе «Параметры командной строки #standalone_system_properties»). Для домена активный «domain.xml» и «host.xml» истории хранятся в «jboss.domain.config.dir/domain_xml_history» и «jboss.domain.config.dir/host_xml_history».

В оставшейся части этого раздела будет рассмотрена только история создания «standalone.xml». Концепции в точности совпадают для «domain.xml» и «host.xml».

В самом «standalone_xml_history» после успешной первой загрузки мы получаем три новых файла:

– «standalone.initial.xml» - в нем содержится исходная конфигурация, которая использовалась при первой успешной загрузке. Этот файл никогда не будет перезаписан. Вы, конечно, можете удалить каталог «history» и любые файлы в нем на любом этапе;

– «standalone.boot.xml» - здесь содержится исходная конфигурация, которая использовалась для последней успешной загрузки сервера. Она перезаписывается при каждой успешной загрузке сервера;

– «standalone.last.xml» - на данном этапе содержимое будет идентично «standalone.boot.xml». Этот файл перезаписывается каждый раз, когда сервер успешно записывает конфигурацию, если произошел непредвиденный сбой при записи конфигурации, этот файл является последней известной успешной записью.

«standalone_xml_history» содержит каталог с именем «current», который должен быть пустым. Теперь, если мы выполним операцию управления, которая изменит модель, например, добавим новое системное свойство с помощью CLI:

```
[standalone@localhost:9990 /] /system-
property=test:add(value="test123"){"outcome" => "success"}
```

Что происходит, так это:

– резервная копия исходного файла конфигурации сохраняется в «standalone_xml_history/current/standalone.v1.xml». При следующем изменении модели будет создан файл с именем «standalone.v2.xml» и т.д. Сохраняются 100 самых последних из этих файлов;

– изменение применяется к исходному файлу конфигурации;

– измененный исходный конфигурационный файл копируется в «standalone.last.xml».

При перезапуске сервера любой существующий каталог «standalone_xml_history/current» перемещается в новую папку с временной меткой в «standalone_xml_history», и создается новая текущая папка. Эти папки с отметками времени хранятся в течение 30 дней.

### 9.7.1 Моментальные снимки

В дополнение к резервным копиям, созданным сервером, как описано выше, вы можете вручную создавать моментальные снимки, которые будут храниться в папке «snapshot» в папке «_xml_history», автоматические резервные копии, описанные выше, подлежат автоматическому хранению дома, поэтому в конечном итоге будут автоматически удалены, с другой стороны, моментальными снимками можно полностью управлять с помощью администратор.

Вы также можете делать свои собственные снимки, используя CLI:

```
[standalone@localhost:9990 /] :take-snapshot
{
   "outcome" => "success",
   "result" => {"name" =>
   "/Users/kabir/WildBoss
   Pro/standalone/configuration/standalone_xml_history/snapshot/2011063
   0-172258657standalone.xml"}
}
```

Вы также можете использовать интерфейс командной строки для составления списка всех моментальных снимков

```
[standalone@localhost:9990 /] :list-snapshots
{
  "outcome" => "success",
  "result" => {
    "directory" =>
    "/Users/kabir/WildBoss
    Pro/standalone/configuration/standalone_xml_history/snapshot",
    "names" => [
       "20110630-165714239standalone.xml",
       "20110630-165821795standalone.xml",
       "20110630-170113581standalone.xml",
       "20110630-171411463standalone.xml",
       "20110630-171908397standalone.xml",
       "20110630-172258657standalone.xml"
    ]
  }
}
```

Чтобы удалить определенный моментальный снимок:

```
[standalone@localhost:9990 /] :delete-snapshot(name="20110630-
165714239standalone.xml")
{"outcome" => "success"}
```

и удалить все моментальные снимки:

```
[standalone@localhost:9990 /] :delete-snapshot(name="all")
{"outcome" => "success"}
```

В режиме домена выполнение операций моментального снимка с корневым узлом будет работать с моделью домена. Чтобы сделать это для модели хоста, вам необходимо перейти к соответствующему хосту:

```
[domain@localhost:9990 /] /host=master:list-snapshots
{
   "outcome" => "success",
   "result" => {
     "domain-results" => {"step-1" => {
       "directory" =>
       "/Users/kabir/WildBoss
       Pro/domain/configuration/host_xml_history/snapshot",
       "names" => [
         "20110630-141129571host.xml",
         "20110630-172522225host.xml"
       ]
     }},
     "server-operations" => undefined
   }
}
```

### 9.7.2 Последующие запуски

Для последующих запусков сервера может оказаться желательным вернуть состояние сервера к одному из ранее известных состояний, для ряда элементов можно использовать сокращенное обращение к файлу:

Таблица 33

| Аббревиатура | Параметр | Описание |
|---|---|---|
| initial | --server-config=initial | при этом сервер будет запущен с использованием начальной конфигурации, которая впервые использовалась для запуска сервера |
| boot | --server-config=boot | при этом будет использоваться конфигурация, полученная при последней успешной загрузке сервера |
| last | --server-config=last | это запустит сервер, используя конфигурацию, сохраненную с момента последнего успешного сохранения |
| v? | --server-config=v? | при этом будет сохранена папка «_xml_history/current» для конфигурации, «где?» это номер используемой резервной копии |
| -? | --server-config=-? | сервер будет запущен после поиска в папке моментальных снимков конфигурации, соответствующей этому префиксу |

В дополнение к этому параметр «--server-config» всегда можно использовать для указания конфигурации, относящейся к «jboss.server.config.dir», и, наконец, если подходящая конфигурация не найдена, будет предпринята попытка найти конфигурацию как абсолютный путь.

## 9.8 История конфигурационных файлов Git

Чтобы улучшить историю исходных конфигурационных файлов, у нас теперь есть встроенная поддержка Git для управления историей конфигурации. Эта функция выходит за

рамки истории исходных конфигурационных файлов, поскольку она также управляет содержимым хранилища контента и всеми файлами конфигурации (такими как свойства). Эта функция работает только для автономных серверов, использующих структуру каталогов по умолчанию.

Как упоминалось в разделе Параметры командной строки, мы поддерживаем использование удаленного репозитория Git для извлечения конфигурации из локального репозитория Git, создания или использования локального репозитория Git. Фактически, если в «jboss.server.base.dir» существует каталог «.git», то автоматически активируется использование Git для управления файлами конфигурации . Каждое изменение содержимого или конфигурации приведет к новой фиксации, если операция выполнена успешно и есть изменения для фиксации. Если есть аутентифицированный пользователь, то это будет сохранен как автор фиксации. Пожалуйста, обратите внимание, что это настоящий Git-репозиторий, поэтому вы можете манипулировать им с помощью собственного Git-клиента.

Теперь, если мы выполним операцию управления, которая изменит модель, например, добавим новое системное свойство с помощью интерфейса командной строки:

```
[standalone@localhost:9990 /] /system-
property=test:add(value="test123"){"outcome" => "success"}
```

Что происходит, так это:

– это изменение будет применено к файлу конфигурации;

– файл конфигурации добавляется в новую фиксацию.

### 9.8.1 Локальный репозиторий Git

Запуск сервера с параметром «--git-repo=local» приведет к созданию репозитория Git, если такового не существует, или к использованию текущего репозитория Git. При запуске локального репозитория Git будет создан файл «.gitignore», который будет добавлен к первоначальной фиксации.

Если добавлен параметр «--git-branch», то репозиторий будет извлечен из предоставленной ветки. Пожалуйста, обратите внимание, что ветка не будет создана автоматически и должна уже существовать в репозитории. По умолчанию, если параметр не указан, будет использоваться главная ветвь.

### 9.8.2 Удаленный репозиторий Git

Если предоставляется удаленный репозиторий Git, сервер попытается извлечь данные из него при загрузке. Если это первый раз, когда мы извлекаем данные, локальные файлы будут удалены, чтобы избежать сбоя при извлечении из-за необходимости перезаписи существующих файлов. Параметром для «--git-repo» может быть URL-адрес или удаленный псевдоним при условии, что вы вручную добавили его в локальную конфигурацию «git».

Если добавлен параметр «--git-branch», то ветвь будет извлечена, в противном случае по умолчанию будет установлено значение «master».

Например, это файл конфигурации «elytron», который можно использовать для подключения к Github с помощью параметра «--git-auth»:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <authentication-client xmlns="urn:elytron:1.0">
   <authentication-rules>
     <rule use-configuration="test-login">
     </rule>
   </authentication-rules>
```

```
    <authentication-configurations>
      <configuration name="test-login">
        <sasl-mechanism-selector selector="BASIC" />
        <set-user-name name="ehsavoie" />
        <credentials>
           <clear-password password="my_api_key" />
        </credentials>
        <set-mechanism-realm name="testRealm" />
      </configuration>
    </authentication-configurations>
  </authentication-client>
</configuration>
```

Пример командной строки для запуска сервера с использованием файла «standalone-full.xml», взятого с Github и прошедшего проверку подлинности с помощью файла конфигурации Elytron «github-WildBoss Pro-config.xml»:

```
./standalone.sh --git-repo=https://github.com/WildBoss Pro/WildBoss
Pro-config.git --git-auth=file:///home/ehsavoie/tmp/github-WildBoss
Pro-config.xml -c standalone-full.xml
```

### 9.8.3 Моментальные снимки

В дополнение к фиксациям, выполненным сервером, как описано выше, вы можете вручную делать снимки, которые будут сохранены в виде тегов в репозитории Git. Вы можете выбрать название тега и сообщение о фиксации, прикрепленное к этому тегу.

Можно делать свои собственные снимки, используя CLI:

```
[standalone@localhost:9990 /] :take-snapshot(name="snapshot",
comment="1st snapshot")
{
  "outcome" => "success",
  "result" => "1st snapshot"
}
```

Также можно использовать интерфейс командной строки для составления списка всех моментальных снимков

```
[standalone@localhost:9990 /] :list-snapshots
{
  "outcome" => "success",
  "result" => {
    "directory" => "",
    "names" => [
      "snapshot : 1st snapshot",
      "refs/tags/snapshot",
      "snapshot2 : 2nd snapshot",
      "refs/tags/snapshot2"
    ]
  }
}
```

Чтобы удалить определенный моментальный снимок:

```
[standalone@localhost:9990 /] :delete-snapshot(name="snapshot2")
{"outcome" => "success"}
```

### 9.8.4 Дистанционный толчок

Возможно, потребуется перенести изменения из вашего репозитория в удаленный репозиторий, чтобы вы могли поделиться ими.

```
[standalone@localhost:9990 /] :publish-
configuration(location="origin")
{"outcome" => "success"}
```

### 9.8.5 Аутентификация по SSH

Пользователи также могут подключаться к SSH-git-серверу. Чтобы подключиться к любому SSH-git-серверу для управления историей ваших файлов конфигурации, вы должны использовать файл конфигурации Elytron, чтобы указать свои учетные данные SSH. В следующем примере показано, как указать URL-адрес по SSH и файл «WildBoss Pro-config.xml» содержащие учетные данные SSH:

```
./standalone.sh --git-repo=git@github.com:WildBoss Pro/WildBoss Pro-
config.git --git-auth=file:///home/user/github-WildBoss Pro-config.xml
```

Есть несколько способов, чтобы указать учетные данные SSH в файле «WildBoss Pro-config.xml»:

– *учетные данные о местоположении ключа SSH*.

Вы можете сослаться на файл, содержащий ваши SSH-ключи, следующим образом:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <authentication-client xmlns="urn:elytron:client:1.6">
   <authentication-rules>
     <rule use-configuration="test-login">
     </rule>
   </authentication-rules>
   <authentication-configurations>
     <configuration name="test-login">
      <credentials>
        <ssh-credential ssh-directory="/home/user/git-persistence/"
        private-key-file="id_ec_test" known-hosts-file="known_hosts">
          <clear-password password="secret"/>
        </ssh-private-key>
      </credentials>
     </configuration>
   </authentication-configurations>
 </authentication-client>
</configuration>
```

Эта конфигурация указывает на то, что закрытый ключ, который будет использоваться для аутентификации по SSH, находится в файле «id_ec_test» в каталоге «/home/user/git-persistence», и для расшифровки ключа требуется ключевая фраза «secret».

SSH - учетные данные принимают следующие атрибуты:

а) «ssh-directory» - путь к каталогу, содержащему файл закрытого ключа и файл известных хостов. Значение по умолчанию – «[user.home]/.ssh»;

б) «private-key-file» - имя файла, содержащего закрытый ключ. По умолчанию используются следующие имена файлов с закрытыми ключами: «id_rsa, id_dsa» и «id_ecdsa»;

в) «known-hosts-file» - имя файла, содержащего известные SSH-хосты, которым вы доверяете. Значение по умолчанию – «known_hosts».

Один из следующих дочерних элементов также может быть использован для указания ключевой фразы, которая будет использоваться для расшифровки закрытого ключа (если применимо):

```
<ssh-credential ...>
<credential-store-reference store="..." alias="..." clear-text="..."
/>
    <clear-password password="..." />
    <masked-password algorithm="..." key-material="..." iteration-
    count="..." salt="..." masked-password="..." initialization-
    vector="..." />
</ssh-credential>
```

– *учетные данные пары ключей*.

Также можно указать свои учетные данные SSH в качестве пары ключей следующим образом:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <authentication-client xmlns="urn:elytron:client:1.6">
  <authentication-rules>
    <rule use-configuration="test-login">
    </rule>
  </authentication-rules>
  <authentication-configurations>
   <configuration name="test-login">
    <credentials>
     <key-pair>
      <openssh-private-key pem="-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABCdRswttV
UNQ6nKb6ojozTGAAAAEAAAAAEAAABoAAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlz
dHAyNTYAAABBBAKxnsRT7n6qJLKoD3mFfAvcH5ZFUyTzJVW8t60pNgNaXO4q5S4qL9yCCZ
cKyg6QtVgRuVxkUSseuR3fiubyTnkAAADQq3vrkvuSfm4n345STr/i/29FZEFUd0qD++B2
ZoWGPKU/xzvxH7S2GxREb5oXcIYO889jY6mdZT8LZm6ZZig3rqoEAqdPyllHmEadb7hY+y
jwcQ4Wr1ekGgVwNHCNu2in3cYXxbrYGMHc33WmdNrbGRDUzK+EEUM2cwUiM7Pkrw5s88Ff
IWI0V+567Ob9LxxIUO/QvSbKMJGbMM4jZ1V9V2Ti/GziGJ107CBudZr/7wNwxIK86BBAEg
      hfnrhYBIaOLrtP8R+96i8iu4iZAvcIbQ==
      -----END OPENSSH PRIVATE KEY-----">
       <clear-password password="secret"/>
      </openssh-private-key>
     </key-pair>
    </credentials>
   </configuration>
  </authentication-configurations>
 </authentication-client>
</configuration>
```

Если ваши известные SSH-хосты отсутствуют в «~/.ssh/known_hosts», вам следует указать ssh-учетные данные вместе с атрибутами «ssh-directory» и «known-hosts-file», чтобы

указать местоположение и имя вашего файла «known-hosts», а также пару учетных данных, связанных с парой ключей.

При указании ключей в формате OpenSSH необходимо указать только закрытый ключ, а открытый ключ будет проанализирован из строки закрытого ключа. При указании пар ключей в формате PKCS необходимо указать как закрытый, так и открытый ключи, используя следующие элементы:

```
<key-pair>
   <private-key-pem>-----BEGIN PRIVATE KEY-----
                    MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgj+To
                    YNaHz/pISg/ZI9BjdhcTre/SJpIxASY19XtOV1ehRANCAASngcxU
                    TBf2atGC5lQWCupsQGRNwwnK6Ww9Xt37SmaHv0bX5n1KnsAal0yk
                    JVKZsD0Z09jVF95jL6udwaKpWQwb
   -----END PRIVATE KEY-----</private-key>
   <public-key-pem>-----BEGIN PUBLIC KEY-----
                    MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEp4HMVEwX9mrRguZU
                    FgrqbEBkTcMJyulsPV7d+0pmh79G1+Z9Sp7AGpdMpCVSmbA9GdPY
                    1RfeYy+rncGiqVkMGw==
   -----END PUBLIC KEY-----</public-key>
</key-pair>
```

При использовании пары учетных данных в формате OpenSSH также можно указать ключевую фразу, которая будет использоваться для расшифровки закрытого ключа:

```
<openssh-private-key pem="...">
   <credential-store-reference store="..." alias="..." clear-
   text="..." />
   <clear-password password="..." />
   <masked-password algorithm="..." key-material="..." iteration-
   count="..." salt="..." masked-password="..." initialization-
   vector="..." />
</ssh-private-key-file>
```

При использовании ключей, отформатированных в формате PKCS, ключи не должны быть зашифрованы парольной фразой;

– *ссылка на хранилище учетных данных*.

Можно указать свои учетные данные SSH в качестве ссылки на запись в хранилище учетных данных. Смотрите: «Добавление учетных данных» в хранилище учетных данных и ссылки на учетные данные, хранящиеся в хранилище учетных данных.

## 9.9 Файл конфигурации (YAML) (экспериментальный))

Чтобы каким-то образом отделить настройку от конфигурации, предоставляемой Galleon, мы хотим поэкспериментировать с новой функцией, в которой эта настройка предоставляется файлами YAML. Поскольку эта функция является «EXPERIMENTAL», мы отключаем ее по умолчанию, и вы не можете рассчитывать на то, что она будет присутствовать или совместима в будущих версиях.

### 9.9.1 Активируйте функцию поддержки YAML

Чтобы включить эту функцию, вам необходимо добавить конфигурацию «ServiceLoader» в модуль «org.jboss.as.controller». Вам необходимо создать следующий файл: «METAINF/services/org.jboss.as.controller.persistence.ConfigurationExtension», содержащее одну строку «**org.jboss.as.controller.persistence.yaml.YamlConfigurationExtension**» в папке «dir» модуля «org.jboss.as.controller».

```
mkdir -p
$WILDBOSS
PRO_HOME/modules/system/layers/base/org/jboss/as/controller/main/dir/ME
TAINF/services/
echo
'org.jboss.as.controller.persistence.yaml.YamlConfigurationExtension' >
$WILDBOSS
PRO_HOME/modules/system/layers/base/org/jboss/as/controller/main/dir/ME
TAINF/services/org.jboss.as.controller.persistence.ConfigurationExtensi
on
```

### 9.9.2 Начиная с файлов YAML

Используя аргумент «--yaml» или «-y», вы можете передать список файлов YAML. Каждый путь должен быть разделен параметром «File.pathSeparator». Это точка с запятой (;) в Windows и двоеточие (:) в операционных системах на базе Mac и Unix. Пути могут быть абсолютными, относительно текущего каталога выполнения или относительно автономного каталога конфигурации.

```
./standalone.sh -y=/home/ehsavoie/dev/WildBoss
Pro/config2.yml:config.yml -c standalonefull.xml
```

### 9.9.3 Что находится в YAML

Корневой узел YAML должен быть «WildBoss Pro-configuration», тогда вы можете использовать дерево моделей для добавления или обновления ресурсов.

Пример файла YAML для определения нового источника данных PostgreSQL:

```
WildBoss Pro-configuration:
 subsystem:
   datasources:
    jdbc-driver:
      postgresql:
        driver-name: postgresql
        driver-xa-datasource-class-name:
        org.postgresql.xa.PGXADataSource
        driver-module-name: org.postgresql.jdbc
    data-source:
    PostgreSQLDS:
      enabled: true
      exception-sorter-class-name:
      org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExcept
      ionSorter
      jndi-name: java:jboss/datasources/PostgreSQLDS
      jta: true
      max-pool-size: 20
      min-pool-size: 0
      connection-url: "jdbc:postgresql://localhost:5432}/demo"
      driver-name: postgresql
      user-name: postgres
      password: postgres
      validate-on-match: true
      background-validation: false
```

```
      background-validation-millis: 10000
      flush-strategy: FailingConnectionOnly
      statistics-enable: false
      stale-connection-checker-class-name:
      org.jboss.jca.adapters.jdbc.extensions.novendor.NullStaleConnect
      ionChecker
      valid-connection-checker-class-name:
      org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidC
      onnectionChecker
      transaction-isolation: TRANSACTION_READ_COMMITTED
```

Мы также предлагаем три операции с использованием тегов:

– **!undefine**: чтобы отменить определение атрибута.

Пример файла YAML для определения уровня консольного регистратора:

```
WildBoss Pro-configuration:
 subsystem:
  logging:
   console-handler:
    CONSOLE:
     level: !undefine
```

– **!remove**: чтобы удалить ресурс.

Пример файла YAML для удаления микропрофиля Small rye из подсистемы JWT:

```
WildBoss Pro-configuration:
 subsystem:
   microprofile-jwt-smallrye: !remove
```

– **!list-add**: чтобы добавить элемент в список (с необязательным индексом).

Пример файла YAML для добавления разрешения на удаленную транзакцию в список разрешений в позиции «0»:

```
WildBoss Pro-configuration:
 subsystem:
  elytron:
   permission-set:
    default-permissions:
     permissions: !list-add
      - class-name:org.WildBoss Pro.transaction.client.
      RemoteTransactionPermission
        module: org.WildBoss Pro.transaction.client
        target-name: "*"
        index: 0
```

# 10 Ссылка на API управления

Этот раздел является подробным описанием API управления WildBoss Pro. Читателям рекомендуется ознакомиться с разделами «Клиенты управления» и «Основные концепции управления» для получения основной справочной информации, а также с разделами «Задачи управления» и «Настройка домена» для получения ключевой информации, ориентированной на решение задач. Этот раздел предназначен для подробного ознакомления с некоторыми ключевыми деталями.

## 10.1 Глобальные операции

API управления WildBoss Pro включает в себя ряд операций, которые применимы к каждому ресурсу.

### 10.1.1 Операция чтения ресурса

Считывает значения атрибутов управляющего ресурса вместе с базовой или полной информацией о любых дочерних ресурсах. Поддерживает следующие параметры, ни один из которых не является обязательным:

– «recursive» – (логическое значение, по умолчанию равно «false») - следует ли включать полную информацию о дочерних ресурсах рекурсивно;

– «recursive-depth» - (int) - глубина, на которую должна быть включена информация о дочерних ресурсах, если значение «recursive» равно «true». Если значение не задано, глубина будет неограниченной, т.е. будут включены все ресурсы-потомки;

– «proxies» - (логическое значение, по умолчанию равно «false») - включать ли удаленные ресурсы в рекурсивный запрос (т.е. ресурсы уровня хоста от подчиненных контроллеров хоста в запрос контроллера домена; использование ресурсов сервера в запросе хоста);

– «include-runtime» - (логическое значение, по умолчанию равно «false») - следует ли включать в ответ атрибуты времени выполнения (т.е. те, значение которых не зависит от постоянной конфигурации);

– «include-defaults» - (логическое значение, по умолчанию равно «false») - следует ли включать в результат значения по умолчанию, не заданные пользователями. Многие атрибуты имеют значения по умолчанию, которые будут использоваться во время выполнения, если пользователи не предоставили явное значение. Если этот параметр равен «false», значение для таких атрибутов в результате будет неопределенным. Если значение равно «true», то результат будет содержать значения по умолчанию для таких параметров.

### 10.1.2 Операция чтения атрибута

Считывает значение отдельного атрибута. Принимает единственный обязательный параметр:

– «name» - (string) - имя атрибута для чтения;

– «include-defaults» - (логическое значение, по умолчанию равно «true») - следует ли включать в результат значения по умолчанию, не заданные пользователями. Многие атрибуты имеют значения по умолчанию, которые будут использоваться во время выполнения, если пользователи не предоставили явное значение. Если этот параметр равен «false», значение для таких атрибутов в результате будет неопределенным. Если значение равно «true», то результат будет содержать значения по умолчанию для таких параметров.

### 10.1.3 Операция записи атрибута

Записывает значение отдельного атрибута. Принимает два обязательных параметра:

– «name» - (string) - имя атрибута для записи;

– «value» - (тип зависит от записываемого атрибута) - новое значение.

### 10.1.4 Операция с неопределенным атрибутом

Присваивает значению отдельного атрибута значение undefined, если такое значение разрешено для атрибута. Операция завершится ошибкой, если значение undefined не разрешено. Принимает единственный обязательный параметр:

– «name» - (string) - имя атрибута для записи.

### 10.1.5 Операция добавления в список

Добавляет элемент к значению атрибута списка, добавляя элемент в конец списка, если только не передан индекс необязательного атрибута:

– «name» - (string) - имя атрибута списка, к которому нужно добавить новое значение;

– «value» - (тип зависит от записываемого элемента) - новый элемент, который будет добавлен к значению атрибута;

– «index» - (int, необязательный) - укажите, в каком месте списка нужно добавить новый элемент. По умолчанию значение не определено, то есть добавить в конце. Индекс равен нулю.

Эта операция завершится ошибкой, если указанный атрибут не является списком.

### 10.1.6 Операция удаления списка

Удаляет элемент из значения атрибута списка, либо элемент с указанным индексом, либо первый элемент, значение которого соответствует указанному значению:

– «name» - (string) - имя атрибута списка, к которому нужно добавить новое значение;

– «value» - (тип зависит от записываемого элемента, необязательно) - элемент, который нужно удалить. Необязательно и игнорируется, если указан индекс;

– «index» - (int, необязательный) - укажите в списке, элемент которого должен быть удален. По умолчанию он не определен, то есть должно быть указано значение.

Эта операция завершится ошибкой, если указанный атрибут не является списком.

### 10.1.7 Операция получения списка

Возвращает один элемент из атрибута списка по его индексу:

– «name» - (string) - имя атрибута списка;

– «index» - (int, требуется) - индекс элемента, который нужно получить из списка.

Эта операция завершится ошибкой, если указанный атрибут не является списком.

### 10.1.8 Операция очистки списка

Очищает атрибут «list». Он отличается от «:undefined-attribute» тем, что приводит к атрибуту типа «list» с «0» элементами, тогда как «:undefined-attribute» приводит к неопределенному значению атрибута:

– «name» - (string) - имя атрибута списка.

Эта операция завершится ошибкой, если указанный атрибут не является списком.

### 10.1.9 Операция «Нанесение на карту»

Добавляет запись пары ключ/значение к значению атрибута карты:

– «name» - (string) - имя атрибута карты, в который будет добавлена новая запись;

– «key» - (string) - ключ новой записи, которая будет добавлена;

– «value» - (тип зависит от записываемой записи) - значение новой записи, которое будет добавлено к значению атрибута.

Эта операция завершится неудачей, если указанный атрибут не является картой.

### 10.1.10 Операция удаления карты

Удаляет запись из значения атрибута карты:

– «name» - (string) - имя атрибута карты, из которого необходимо удалить новую запись;

– «key» - (string) - ключ записи, которую необходимо удалить.

Эта операция завершится неудачей, если указанный атрибут не является картой.

### 10.1.11 Операция получения карты

Возвращает значение одной записи из атрибута карты:

– «name» - (string) - имя атрибута карты;

– «key» - (string) - ключ к записи.

Эта операция завершится неудачей, если указанный атрибут не является картой.

### 10.1.12 Операция очистки карты

Очищает атрибут «map». Он отличается от «:undefined-attribute» тем, что приводит к атрибуту типа «map» с «0» записями, тогда как «:undefined-attribute» приводит к неопределенному значению атрибута:

– «name» - (string) - имя атрибута карты.

Эта операция завершится неудачей, если указанный атрибут не является картой.

### 10.1.13 Операция чтения описания ресурса

Возвращает описание атрибутов ресурса, типов дочерних элементов и, при необходимости, операций. Поддерживает следующие параметры, ни один из которых не является обязательным:

– «recursive» - (логическое значение, по умолчанию равно «false») - следует ли включать информацию о дочерних ресурсах рекурсивно;

– «proxies» - (логическое значение, по умолчанию равно «false») - включать ли удаленные ресурсы в рекурсивный запрос (т.е. ресурсы уровня хоста от подчиненных контроллеров хоста в запрос контроллера домена; использование ресурсов сервера в запросе хоста);

– «operations» - (логическое значение, по умолчанию равно «false») - следует ли включать описания операций ресурса;

– «inherited» - (логическое значение, по умолчанию равно «false») - если значение «operations» равно «true», следует ли включать описания операций, унаследованных от ресурсов более высокого уровня. Глобальные операции, описанные в этом разделе, сами по себе унаследованы от корневого ресурса, поэтому основной целью установки значения «inherited» в значение «false» является исключение описаний глобальных операций из выходных данных.

Более подробную информацию о результате этой операции смотрите в описании модели управления.

### 10.1.14 Операция чтения имен операций

Возвращает список названий всех операций, поддерживаемых ресурсом. Не принимает параметров.

### 10.1.15 Операция чтения-описания операции

Возвращает описание операции, а также подробные сведения о типах ее параметров и возвращаемом значении. Принимает единственный обязательный параметр:

– «name» - (string) - название операции.

Более подробную информацию о результате этой операции смотрите в описании модели управления.

### 10.1.16 Операция чтения дочерних типов

Возвращает список типов дочерних ресурсов, поддерживаемых данным ресурсом. Принимает два необязательных параметра:

– «include-aliases» - (логическое значение, по умолчанию равно «false») - следует ли включать в ответ дочерние псевдонимы (т.е. те, которые являются псевдонимами других подресурсов);

– «include-singletons» - (логическое значение, по умолчанию равно «false») - следует ли включать дочерние элементы «singleton» (т.е. дочерние элементы, которые действуют как совокупность ресурсов и зарегистрированы с подстановочным именем) в ответ на обсуждение «WildBoss Pro-dev» по этой теме.

### 10.1.17 Операция чтения дочерних имен

Возвращает список имен всех дочерних ресурсов заданного типа. Принимает единственный обязательный параметр:

– «child-type» - (string) - название типа.

### 10.1.18 Операция «Чтение дочерних ресурсов»

Возвращает информацию обо всех дочерних элементах ресурса, которые относятся к заданному типу. Для каждого дочернего ресурса возвращаемая информация эквивалентна выполнению операции чтения ресурса для этого ресурса. Принимает следующие параметры, из которых требуется только «\{{дочерний тип}»:

– «child-type» - (string) - имя типа дочернего ресурса;

– «recursive» - (логическое значение, по умолчанию равно «false») - следует ли включать полную информацию о дочерних ресурсах рекурсивно;

– «recursive-depth» - (int) - глубина, на которую должна быть включена информация о дочерних ресурсах при рекурсивном использовании, равна «\{{true}». Если значение не задано, глубина будет неограниченной, т.е. будут включены все ресурсы-потомки;

– «proxies» - (логическое значение, по умолчанию равно «false») - включать ли удаленные ресурсы в рекурсивный запрос (т.е. ресурсы уровня хоста от подчиненных контроллеров хоста в запрос контроллера домена; использование ресурсов сервера в запросе хоста);

– «include-runtime» - (логическое значение, по умолчанию равно «false») - следует ли включать в ответ атрибуты времени выполнения (т.е. те , значение которых не зависит от постоянной конфигурации);

– «include-defaults» - (логическое значение, по умолчанию равно «true») - следует ли включать в результат значения по умолчанию, не заданные пользователями. Многие атрибуты имеют значения по умолчанию, которые будут использоваться во время выполнения, если пользователи не предоставили явное значение. Если этот параметр равен «false», значение для таких атрибутов в результате будет неопределенным. Если значение равно «true», то результат будет содержать значения по умолчанию для таких параметров.

### 10.1.19 Операция чтения группы атрибутов

Возвращает список атрибутов определенного типа для заданного имени группы атрибутов. Для каждого атрибута возвращаемая информация эквивалентна выполнению операции чтения атрибута этого ресурса. Принимает следующие параметры, из которых требуется только «\{{имя}»:

– «name» - (string) - имя группы атрибутов для чтения;

– «include-defaults» - (логическое значение, по умолчанию равно «true») - следует ли включать в результат значения по умолчанию, не заданные пользователями. Многие атрибуты имеют значения по умолчанию, которые будут использоваться во время выполнения, если пользователи не предоставили явное значение. Если этот параметр равен «false», значение для таких атрибутов в результате будет неопределенным. Если значение равно «true», то результат будет содержать значения по умолчанию для таких параметров;

– «include-runtime» - (логическое значение, по умолчанию равно «false») - следует ли включать в ответ атрибуты времени выполнения (т.е. те, значение которых не зависит от постоянной конфигурации);

– «include-aliases» - (логическое значение, по умолчанию равно «false») - следует ли включать в ответ атрибуты псевдонимов (т.е. те, которые являются псевдонимами других атрибутов).

### 10.1.20 Операция чтения имен групп атрибутов

Возвращает список имен групп атрибутов для данного типа. Не принимает параметров.

### 10.1.21 Стандартные операции

Помимо глобальных операций, описанных выше, по соглашению почти каждый ресурс должен предоставлять доступ к операциям добавления и удаления. Исключением из этого соглашения являются корневой ресурс и ресурсы, которые не хранят постоянную конфигурацию и создаются динамически во время выполнения (например, ресурсы, представляющие «mbeans» платформы виртуальной машины JAVA, или ресурсы, представляющие аспекты текущего состояния развертывания).

*Операция добавления* - операция, которая создает новый ресурс, должна называться «add». Операция может принимать ноль или более параметров; что это за параметры, зависит от создаваемого ресурса.

*Операция удаления* - Операция, которая удаляет существующий ресурс, должна называться «remove». Операция не должна содержать параметров.

## 10.2 Детализированное управление и библиотека «jboss-dmr»

Модель управления, представленная WildBoss Pro, очень масштабна и сложна. Существуют десятки, возможно, сотни задействованных логических концепций – хосты, группы серверов, подсистемы, источники данных, веб –соединители и так далее, - каждая из которых в классическом объектно-ориентированном дизайне API может быть представлена типом Java (т.е. классом или интерфейсом Java). Однако основной целью разработки собственного API управления WildBoss Pro было обеспечить, чтобы клиенты, созданные для использования этого API, имели как можно меньше зависимостей во время компиляции и выполнения от классов, предоставляемых JBoss, и чтобы API, предоставляемый этими библиотеками, должен быть мощным, но в то же время простым и стабильным. Клиент управления, работающий с библиотеками управления, созданными для более ранней версии WildBoss Pro, должен по-прежнему работать, если используется для управления доменом более поздней версии. Клиентские библиотеки управления должны быть совместимы с прямой версией.

Крайне маловероятно, что API, состоящий из сотен типов Java, может быть совместим в прямом эфире. Вместо этого, WildBoss Pro management API является детализированным API. Детализированный API похож на кофе без кофеина – в нем все еще есть немного кофеина, но его недостаточно, чтобы не давать вам спать по ночам. В API управления WildBoss Pro по-прежнему есть несколько типов Java (невозможно, чтобы в библиотеке Java не было типов!) но этого недостаточно, чтобы заставлять вас (или нас) не спать по ночам, беспокоясь о том, что ваши управляющие клиенты не будут совместимы с «forward».

Детализированный API позволяет создавать сколь угодно сложные структуры данных, используя небольшое количество типов Java. Все значения параметров и возвращаемые значения в API выражаются с использованием этих нескольких типов. В идеале большинство типов являются базовыми типами JDK, такими как «java.lang.String», «java.lang.Integer» и т.д. В дополнение к базовым типам JDK, детализированный API управления WildBoss Pro использует небольшую библиотеку под названием «jboss-dmr». Цель этого раздела - предоставить общий обзор библиотеки «jboss-dmr».

Даже если вы не используете «jboss-dmr» напрямую (вероятно, это касается всех, кроме нескольких пользователей), некоторая информация из этого раздела может оказаться полезной. Когда вы вызываете операции с использованием интерфейса командной строки сервера приложений, возвращаемые значения представляют собой просто текстовое представление jbossdmr ModelNode. Если ваши CLI-команды требуют сложных значений параметров, вы можете сами написать текстовое представление узла модели. И если вы используете HTTP management API, все тексты ответов, а также текст запроса для любой публикации будут представлять собой JSON-представление ModelNode.

Исходный код для «jboss-dmr» доступен на Github. Координаты «maven» для выпуска «jboss-dmr» - это «org.jboss.jboss-dmr:jboss-dmr».

### 10.2.1 Узел модели и тип модели

Публичный API, предоставляемый «jboss-dmr», очень прост: всего три класса, один из которых является «enum»!

Основным классом является «org.jboss.dmr.ModelNode». Узел модели — это, по сути, просто оболочка для некоторого значения; обычно это значение имеет некоторый базовый тип JDK. Узел модели предоставляет метод GetType(). Этот метод возвращает значение типа «org.jboss.dmr.ModelType», которое является перечислением всех допустимых типов значений. И это 95% общедоступного API; класс и перечисление. (Ниже мы перейдем к третьему классу, собственности).

*Манипулирование узлами базовой модели* - чтобы проиллюстрировать, как работать с ModelNode s, мы воспользуемся библиотекой сценариев «Beanshell». Здесь мы не будем вдаваться в подробности о «beanshell»; это простой и интуитивно понятный инструмент, и, надеюсь, следующие примеры будут такими же.

Начнем с запуска интерпретатора «beanshell», используя библиотеку «jboss-dmr», доступную в пути к классам. Затем мы попросим «beanshell» импортировать все классы «jboss-dmr», чтобы они были доступны для использования:

```
$ java -cp bsh-2.0b4.jar:jboss-dmr-1.0.0.Final.jar bsh.Interpreter
BeanShell 2.0b4 - by Pat Niemeyer (pat@pat.net)
bsh % import org.jboss.dmr.*;
bsh %
```

Затем создайте узел модели и используйте функцию печати «beanshel»l для вывода его типа:

```
bsh % ModelNode node = new ModelNode();
bsh % print(node.getType());
UNDEFINED
```

Новый узел модели не имеет сохраненного значения, поэтому его тип – «ModelType.UNDEFINED».

Используйте один из перегруженных вариантов метода set для присвоения значения узлу:

```
bsh % node.set(1);
bsh % print(node.getType());
INT
bsh % node.set(true);
bsh % print(node.getType());
BOOLEAN
bsh % node.set("Hello, world");
bsh % print(node.getType());
STRING
```

Используйте один из методов AsXXX() для получения значения:

```
bsh % node.set(2);
bsh % print(node.asInt());
2
bsh % node.set("A string");
bsh % print(node.asString());
A string
```

Узел модели попытается выполнить преобразование типов, когда вы вызовете методы AsXXX:

```
bsh % node.set(1);
bsh % print(node.asString());
1
bsh % print(node.asBoolean());
true
bsh % node.set(0);
bsh % print(node.asBoolean());
false
bsh % node.set("true");
bsh % print(node.asBoolean());
true
```

Не все преобразования типов возможны:

```
bsh % node.set("A string");
bsh % print(node.asInt());
// Error: // Uncaught Exception: Method Invocation node.asInt : at
Line: 20 : in file:<unknown file> : node .asInt ( )
Target exception: java.lang.NumberFormatException: For input string:
"A string"
java.lang.NumberFormatException: For input string: "A string"
at java.lang.NumberFormatException.forInputString
(NumberFormatException.java:48)
at java.lang.Integer.parseInt(Integer.java:449)
at java.lang.Integer.parseInt(Integer.java:499)
at org.jboss.dmr.StringModelValue.asInt(StringModelValue.java:61)
at org.jboss.dmr.ModelNode.asInt(ModelNode.java:117)
    ....
```

Метод Model Node.GetType() можно использовать, чтобы убедиться, что узел имеет ожидаемый тип значения, прежде чем пытаться преобразовать тип.

Один из вариантов «set» принимает другой узел модели в качестве аргумента. Значение переданного узла «in» копируется, поэтому между двумя узлами модели нет общего состояния:

```
bsh % node.set("A string");
bsh % ModelNode another = new ModelNode();
bsh % another.set(node);
bsh % print(another.asString());
A string
bsh % node.set("changed");
bsh % print(node.asString());
changed
bsh % print(another.asString());
 A string
```

Узел модели может быть клонирован. Опять же, между исходным узлом и его клоном нет общего состояния:

```
bsh % clone.protect();
bsh % clone.set("A different string");
// Error: // Uncaught Exception: Method Invocation clone.set : at
Line: 15 : in file:<unknown file> : clone .set ("A different string")
Target exception: java.lang.UnsupportedOperationException
java.lang.UnsupportedOperationException
   at org.jboss.dmr.ModelNode.checkProtect(ModelNode.java:1441)
   at org.jboss.dmr.ModelNode.set(ModelNode.java:351)
    ....
```

*Списки* - приведенные выше примеры не особенно интересны; если все, что мы можем сделать с ModelNode, — это обернуть простой примитив Java, то какой в этом смысл? Однако значение ModelNode может быть более сложным, чем у простого примитива, и, используя эти более сложные типы, мы можем создавать сложные структуры данных. Первый более сложный тип — это Model Type.LIST.

Используйте методы добавления, чтобы инициализировать значение узла в виде списка и добавить в этот список:

```
bsh % ModelNode list = new ModelNode();
bsh % list.add(5);
bsh % list.add(10);
bsh % print(list.getType());
LIST
```

Используйте asInt(), чтобы определить размер списка:

```
bsh % print(list.asInt());
2
```

Используйте перегруженный вариант метода «get», который использует параметр «int» для извлечения элемента. Элемент возвращается как узел модели:

```
bsh % ModelNode child = list.get(1);
bsh % print(child.asInt());
10
```

Не обязательно, чтобы все элементы в списке были одного типа:

```
bsh % list.add("A string");
bsh % print(list.get(1).getType());
INT
bsh % print(list.get(2).getType());
STRING
```

Вот одна из самых сложных особенностей «jboss-dmr»: методы «get» на самом деле изменяют состояние; они не «доступны только для чтения». Например, вызов «get» с индексом, который еще не существует в списке, фактически создаст дочерний элемент типа ModelType.UNDEFINED для этого индекса (и создаст НЕОПРЕДЕЛЕННЫЕ дочерние элементы для любых промежуточных индексов).

```
bsh % ModelNode four = list.get(4);
bsh % print(four.getType());
UNDEFINED
bsh % print(list.asInt());
6
```

Поскольку вызов «get» всегда возвращает узел модели и никогда не возвращает значение «null», манипулировать возвращаемым значением безопасно:

```
bsh % list.get(5).set(30);
bsh % print(list.get(5).asInt());
30
```

В приведенном выше примере это не так интересно, но позже мы рассмотрим узел типа ModelType.OBJECT мы увидим, как такая цепочка методов может позволить вам создавать довольно сложные структуры данных с минимумом кода.

Используйте метод asList(), чтобы получить список <ModelNode> дочерних элементов:

```
bsh % for (ModelNode element : list.asList()) {
print(element.getType());
}
INT
INT
STRING
UNDEFINED
UNDEFINED
INT
```

Методы asString() и toString() предоставляют текстовые представления типа модели в несколько ином формате «ModelType.LIST»:

```
bsh % print(list.asString());
[5,10,"A string",undefined,undefined,30]
bsh % print(list.toString());
[
   5,
   10,
   "A string",
   undefined,
   undefined,
   30
]
```

Наконец, если вы ранее использовали «set» для присвоения значения узла какому-либо типу, отличному от списка, вы не сможете использовать метод «add»:

```
bsh % node.add(5);
// Error: // Uncaught Exception: Method Invocation node.add : at Line:
18 : in file:
<unknown file> : node .add ( 5 )
Target exception: java.lang.IllegalArgumentException
java.lang.IllegalArgumentException
 at org.jboss.dmr.ModelValue.addChild(ModelValue.java:120)
 at org.jboss.dmr.ModelNode.add(ModelNode.java:1007)
 at org.jboss.dmr.ModelNode.add(ModelNode.java:761)
   ...
```

Однако вы можете использовать метод setEmptyList(), чтобы изменить тип узла, а затем использовать «add»:

```
bsh % node.setEmptyList();
bsh % node.add(5);
bsh % print(node.toString());
[5]
```

*Свойства* - третьим открытым классом в библиотеке «jboss-dmr» является «org.jboss.dmr.Property». Свойство — это «String ⇒ModelNode» кортеж.

```
bsh % Property prop = new Property("stuff", list);
bsh % print(prop.toString());
org.jboss.dmr.Property@79a5f739
bsh % print(prop.getName());
stuff
bsh % print(prop.getValue());
[
  5,
  10,
  "A string",
  undefined,
  undefined,
  30
]
```

Это свойство может быть передано в ModelNode.set:

```
bsh % node.set(prop);
bsh % print(node.getType());
PROPERTY
```

Текстовый формат для ModelType.PROPERTY является:

```
bsh % print(node.toString());
("stuff" => [
  5,
  10,
  "A string",
  undefined,
  undefined,
  30
])
```

Прямое создание экземпляра свойства через его конструктор встречается нечасто. Чаще всего используется один из двух вариантов аргументов ModelNode.add или ModelNode.set. Первым аргументом является имя свойства:

```
bsh % ModelNode simpleProp = new ModelNode();
bsh % simpleProp.set("enabled", true);
bsh % print(simpleProp.toString());
("enabled" => true)
bsh % print(simpleProp.getType());
PROPERTY
bsh % ModelNode propList = new ModelNode();
bsh % propList.add("min", 1);
bsh % propList.add("max", 10);
bsh % print(propList.toString());
[
  ("min" => 1),
  ("max" => 10)
]
bsh % print(propList.getType());
LIST
bsh % print(propList.get(0).getType());
PROPERTY
```

Метод asPropertyList() обеспечивает простой доступ к List<Property>:

```
bsh % for (Property prop : propList.asPropertyList()) {
print(prop.getName() + " = " + prop.getValue());
}
min = 1
max = 10
```

**ModelType.OBJECT** - самым мощным и наиболее часто используемым типом комплексного значения в «jboss-dmr» является ModelType.OBJECT. «ModelNode», значение которого равно типу «ModelType.OBJECT» внутренне поддерживает «Map<String», «ModelNode».

Используйте вариант метода «get», который принимает строковый аргумент, чтобы добавить запись на карту. Если запись с указанным именем не существует, добавляется новая запись, значение которой является ModelType.UNDEFINED узлом. Возвращается узел:

```
bsh % ModelNode range = new ModelNode();
bsh % ModelNode min = range.get("min");
bsh % print(range.toString());
{"min" => undefined}
bsh % min.set(2);
bsh % print(range.toString());
{"min" => 2}
```

Опять же, важно помнить, что операция «get» может изменить состояние узла модели, добавив новую запись. Это операция, доступная не только для чтения.

Поскольку «get» никогда не вернет значение «null», обычным способом является использование цепочки методов для создания пары ключ/значение:

```
bsh % range.get("max").set(10);
bsh % print(range.toString());
{
```

```
   "min" => 2,
   "max" => 10
}
```

Вызов «get», передающий уже существующий ключ, конечно, вернет тот же узел модели, который был возвращен при первом вызове «get» с этим ключом:

```
bsh % print(min == range.get("min"));
true
```

В «get» можно передать несколько параметров. Это простой способ просмотра дерева, состоящего из узлов ModelType.OBJECT узлов. Опять же, «get» может изменить узел, на котором он вызывается; например, он фактически создаст дерево, если узлы не существуют. В следующем примере используется обходной путь, позволяющий заставить «beanshell» обрабатывать перегруженный метод «get», который принимает переменное количество аргументов:

```
bsh % String[] varargs = { "US", "Missouri", "St. Louis" };
bsh % salesTerritories.get(varargs).set("Brian");
bsh % print(salesTerritories.toString());
{"US" => {"Missouri" => {"St. Louis" => "Brian"}}}
```

Обычный синтаксис был бы следующим:

```
salesTerritories.get("US", "Missouri", "St. Louis").set("Brian");
```

Пары ключ/значение на карте доступны в виде «List<Property»:

```
bsh % for (Property prop : range.asPropertyList()) {
print(prop.getName() + " = " + prop.getValue());
}
min = 2
```

Семантика вспомогательной карты в узле типа «ModelType.OBJECT» являются объекты LinkedHashMap. Карта запоминает порядок, в котором добавляются пары ключ/значение. Это актуально при переборе пар после вызова функции asPropertyList() и для управления порядком, в котором пары ключ/значение отображаются в выходных данных функции toString().

Поскольку метод «get» фактически изменяет состояние узла, если заданный ключ не существует, ModelNode предоставляет несколько методов, позволяющих проверить, есть ли там запись. Метод «has» просто выполняет это:

```
bsh % print(range.has("unit"));
false
bsh % print(range.has("min"));
true
```

Очень часто необходимо не только знать, существует ли пара ключ/значение, но и определено ли значение (т.е. не ModelType.UNDEFINED. Этот вид проверки аналогичен проверке того, имеет ли поле в классе Java значение «null». Параметр «hasDefined» позволяет сделать это:

```
bsh % print(range.hasDefined("unit"));
false
bsh % // Establish an undefined child 'unit';
bsh % range.get("unit");
bsh % print(range.toString());
{
   "min" => 2,
   "max" => 10,
   "unit" => undefined
}
bsh % print(range.hasDefined("unit"));
false
bsh % range.get("unit").set("meters");
bsh % print(range.hasDefined("unit"));
true
```

**ModelType.EXPRESSION** - значение типа ModelType.EXPRESSION хранится в виде строки, но позже может быть преобразовано в другое значение. Строка имеет специальный синтаксис, который должен быть знаком тем, кто использовал функцию подстановки системных свойств в предыдущих версиях JBoss AS.

```
[<prefix>][${<system-property-name>[:<default-value>]}][<suffix>]*
```

Например:

```
${queue.length}
http://${host}
http://${host:localhost}:${port:8080}/index.html
```

Используйте метод «setExpression», чтобы задать для значения узла значение типа «expression»:

```
bsh % ModelNode expression = new ModelNode();
bsh % expression.setExpression("${queue.length}");
bsh % print(expression.getType());
EXPRESSION
```

Вызов функции asString() возвращает ту же строку, которая была введена:

```
bsh % print(expression.asString());
${queue.length}
```

Однако вызов функции toString() сообщает вам, что значение этого узла не относится к типу ModelType.STRING:

```
bsh % print(expression.toString());
expression "${queue.length}"
```

Когда вызывается операция разрешения, строка анализируется, и все встроенные системные свойства сопоставляются с текущими значениями системных свойств виртуальной машины JAVA. Возвращается новый узел модели, значение которого является разрешенной строкой:

```
bsh % System.setProperty("queue.length", "10");
bsh % ModelNode resolved = expression.resolve();
bsh % print(resolved.asInt());
10
```

Обратите внимание, что типом ModelNode, возвращаемого функцией resolve(), является ModelType.STRING:

```
bsh % print(resolved.getType());
STRING
```

Вызов resolved.async() в предыдущем примере сработал только потому, что строка «10» оказалась преобразуемой в «int 10».

Вызов функции resolve() не влияет на значение узла, на котором вызывается метод:

```
bsh % resolved = expression.resolve();
bsh % print(resolved.toString());
"10"
bsh % print(expression.toString());
expression "${queue.length}"
```

Если выражение не может быть разрешено, «resolve» просто использует исходную строку. Строка может содержать более одной замены системного свойства:

```
bsh % expression.setExpression("http://${host}:${port}/index.html");
bsh % resolved = expression.resolve();
bsh % print(resolved.asString());
http://${host}:${port}/index.html
```

При желании выражение может содержать значение по умолчанию, отделенное от имени системного свойства двоеточием:

```
bsh %
expression.setExpression("http://${host:localhost}:${port:8080}/index.h
tml");
bsh % resolved = expression.resolve();
bsh % print(resolved.asString());
http://localhost:8080/index.html
```

На самом деле включение подстановки системного свойства в выражение не требуется:

```
bsh % expression.setExpression("no system property");
bsh % resolved = expression.resolve();
bsh % print(resolved.asString());
no system property
bsh % print(expression.toString());
expression "no system property"
```

Метод «resolve» работает и на узлах других типов; он возвращает копию без каких-либо попыток реального разрешения:

```
bsh % ModelNode basic = new ModelNode();
bsh % basic.set(10);
bsh % resolved = basic.resolve();
bsh % print(resolved.getType());
```

```
INT
bsh % resolved.set(5);
bsh % print(resolved.asInt());
5
bsh % print(basic.asInt());
10
```

В дополнение к системным свойствам, в приведенных выше примерах мы также поддерживаем замену из переменных среды. Более подробное описание того, как это работает на практике, смотрите в подразделе «Разрешение выражений».

**ModelType.TYPE -** также можно передать одно из значений перечисления типа модели для установки:

```
bsh % ModelNode type = new ModelNode();
bsh % type.set(ModelType.LIST);
bsh % print(type.getType());
TYPE
bsh % print(type.toString());
LIST
```

Это полезно при использовании структуры данных узла модели для описания структуры данных другого узла модели.

**Полный список типов ModelNode.**

BIG_DECIMAL
BIG_INTEGER
BOOLEAN
BYTES
DOUBLE
EXPRESSION
INT
LIST
LONG
OBJECT
PROPERTY
STRING
TYPE
UNDEFINED

**Текстовое представление ModelNode**

ЗАДАЧА (TODO) – задокументировать грамматику.

**JSON-представление ModelNode**

ЗАДАЧА (TODO) – задокументировать грамматику.

### 10.3 Описание модели управления

Подробное описание ресурсов, атрибутов и операций, составляющих модель управления, предоставляемую отдельным экземпляром WildBoss Pro или любым контроллером домена или подчиненным хостом. Процесс контроллера может быть запрошен с помощью операций «read-resource-description», «read-operation-names», «read operation-description» и «read-child-types», описанных в разделе «Глобальные операции». В этом разделе мы подробно расскажем о том, что включено в эти описания.

### 10.3.1 Описание ресурсов, управляемых WildBoss Pro

Все части модели управления, предоставляемые WildBoss Pro, могут быть адресованы с помощью упорядоченного списка пар ключ/значение. Для каждого адресуемого ресурса управления будет доступна следующая описательная информация:

– «description» - (String) - текстовое описание этой части модели;

– «min-occurs» - (int, либо «0», либо «1») - минимальное количество ресурсов этого типа, которое должно существовать в допустимой модели. Если его нет, значение по умолчанию равно «0»;

– «max-occurs» - (int) - максимальное количество ресурсов этого типа, которое может существовать в допустимой модели. Если нет, значение по умолчанию зависит от значения конечной пары ключ/значение в адресе описываемого ресурса. Если это значение равно «*», то значение по умолчанию равно целому Integer.MAX_VALUE значение, т.е. ограничений нет. Если это значение является какой-либо другой строкой, то значение по умолчанию равно «1»;

– «attributes» - (сопоставление строки (имени атрибута) со сложной структурой) - атрибуты конфигурации, доступные в этой части модели. Представление каждого атрибута смотрите в разделе «Описание атрибута»;

– «operations» - (сопоставление строки (имени атрибута) со сложной структурой) - операции, которые могут быть направлены по этому адресу. Описание каждой операции приведено в разделе «Описание операции»;

– «children» - (сопоставление строки (имени атрибута) со сложной структурой) - связь этой части модели с другими адресуемыми частями модели. Представление каждой дочерней связи смотрите в разделе «Описание родительских и дочерних связей»;

– «head-comment-allowed» - (boolean) - этот ключ описания предназначен для возможного использования в будущем;

– «tail-comment-allowed» - (boolean) - этот ключ описания предназначен для возможного использования в будущем.

Например:

```
{
  "description => "A manageable resource",
  "tail-comment-allowed" => false,
  "attributes" => {
    "foo" => {
       .... details of attribute foo
    }
  },
  "operations" => {
    "start" => {
       .... details of the start operation
    }
  },
  "children" => {
    "bar" => {
       .... details of the relationship with children of type "bar"
    }
  }
}
```

*Описание атрибута*

Атрибут — это часть модели управления, к которой нельзя обратиться напрямую. Вместо этого, по сути, это свойство адресуемого ресурса управления. Для каждого атрибута в модели будет доступна следующая описательная информация:

– «description» - (String) - текстовое описание атрибута;

– «type» - (org.jboss.dmr.ModelType) - тип значения атрибута. Одно из значений перечисления «BIG_DECIMAL», «BIG_INTEGER», «BOOLEAN», «BYTES», «DOUBLE», «INT», «LIST», «LONG», «OBJECT», «PROPERTY», «STRING». Большинство из них не требуют пояснений. «OBJECT» будет представлен в детализированной модели как сопоставление строковых ключей со значениями некоторого другого допустимого типа, концептуально аналогичного «javax.management.openmbean.CompositeData». «PROPERTY» — это единственная пара ключ/значение, где ключ представляет собой строку, а значение имеет какой-либо другой допустимый тип;

– «value-type» - (ModelType или сложная структура) - присутствует только в том случае, если тип – «LIST» или «OBJECT». Если все элементы в «LIST» или все значения типа «OBJECT» имеют один и тот же тип, то это будет одно из перечислений ModelType «BIG_DECIMAL», «BIG_INTEGER», «BOOLEAN», «BYTES», «DOUBLE», «INT», «LONG», «STRING». В противном случае «value-type» будет содержать подробную информацию о структуре значения атрибута, перечисляя поля значения и тип их значений. Итак, атрибут с типом «LIST» и «value-type» значение «ModelType.STRING» аналогична списку Java<String>, а строка «value-type» со значением «ModelType.INT» аналогична списку Java<Integer>. Атрибут с типом «OBJECT» и значением типа «ModelType.STRING» аналогична Java Map<String, String>.. Атрибут с типом «OBJECT» и «value-type», значение которого не относится к типу «ModelType», представляет собой полностью определенный сложный объект с описанными правовыми полями объекта и их значениями;

– «expressions-allowed» - (boolean) - указывает, может ли значение атрибута иметь тип ModelType.EXPRESSION, а не его стандартный тип (см. «type» и «value-type» выше для обсуждения стандартного типа атрибута.) Значение «ModelType.EXPRESSION» содержит выражение для подстановки системного свойства или переменной среды, которое сервер разрешит с помощью сопоставления системных свойств на стороне сервера перед использованием значения. Например, атрибут с именем «max-threads» может иметь значение выражения, равное «${example.pool.max-threads:10}», а не просто «10». По умолчанию значение, если оно отсутствует, равно «false». Более подробное описание смотрите в подразделе «Разрешение выражений»;

– «required» - (boolean) - указывает, может ли значение атрибута иметь тип «ModelType.EXPRESSION», а не его стандартный тип (см. «type» и «value-type» выше для обсуждения стандартного типа атрибута.) Значение типа «ModelType.EXPRESSION» содержит выражение подстановки системного свойства или переменной среды, которое сервер разрешит с помощью сопоставления системных свойств на стороне сервера перед использованием значения. Например, атрибут с именем «max-threads» может иметь значение выражения, равное «${example.pool.max-threads:10}», а не просто «10». По умолчанию значение, если оно отсутствует, равно «false». Более подробное описание смотрите в подразделе «Разрешение выражений»;

– «required» - (boolean) – «true», если атрибут должен иметь определенное значение в представлении своей части модели, если только не определен другой атрибут, включенный в список альтернатив; «false», если он может быть неопределенным (подразумевая нулевое значение) даже при отсутствии альтернатив. Если его нет, по умолчанию используется значение «true»;

– «nillable» - (boolean) - значение равно «true», если атрибут может не иметь определенного значения в представлении своей части модели. Атрибут, значение которого может быть равно нулю, может быть, не определен либо потому, что он не требуется, либо потому, что он требуется, но имеет альтернативы, и одна из альтернатив определена;

– «storage» - (String) - либо «configuration», либо «runtime». Если «configuration», то значение атрибута сохраняется как часть постоянной конфигурации (например, в «domain.xml», «host.xml» или «standalone.xml».) Если «runtime», то значение атрибута не сохраняется в постоянной конфигурации; значение существует только до тех пор, пока ресурс запущен;

– «access-type» - (String) - один из вариантов "read-only " (только для чтения), «read-write» (для чтения-записи) или «metric» (метрика). Может ли значение атрибута быть записано или только прочитано. "Метрика" — это атрибут, доступный только для чтения, значение которого не сохраняется в постоянной конфигурации и может изменяться в зависимости от активности на сервере. Если атрибут является «read-write», ресурс предоставит операцию с именем «write-attribute», параметр «name» которой будет принимать имя этого атрибута, а параметр «value» будет принимать допустимое значение для этого атрибута. Эта операция будет стандартным средством обновления значения этого атрибута;

– «restart-required» - (String) - один из «no-services», «all-services», «resource-services» или «jvm». Относится только к атрибутам, тип доступа к которым «read-write». Указывает, требуется ли для выполнения операции «write-attribute», параметр «name» которой определяет этот атрибут, перезапуск служб (или всей виртуальной машины JVM), чтобы изменение вступило в силу во время выполнения. Смотрите обсуждение этого «Применение обновлений» к службам среды выполнения показано ниже. Значение по умолчанию – «no-services»;

– «default» - значение по умолчанию для атрибута, который будет использоваться в службах «runtime services», если этот атрибут явно не определен и другие атрибуты, перечисленные в качестве альтернативных, не определены;

– «alternatives» - (List of string) - указывает на исключительную связь между атрибутами. Если этот атрибут определен, другие атрибуты, перечисленные в значении этого дескриптора, должны быть неопределенными, даже если их обязательный дескриптор указывает «true»; т.е. наличие этого атрибута удовлетворяет требованию. Обратите внимание, что атрибут, который явно не настроен, но имеет значение по умолчанию, по-прежнему считается не определенным для проверки того, не нарушена ли исключительная связь. Значение по умолчанию не определено, т.е. это не относится к большинству атрибутов;

– «requires» - (List of string) - указывает, что если этот атрибут имеет значение (отличное от «undefined»), то другие атрибуты, перечисленные в значении этого дескриптора, также должны иметь значение, даже если их обязательный дескриптор содержит значение «false». Обычно это используется в сочетании с альтернативными вариантами. Например, атрибуты «a» и «b» обязательны, но являются альтернативами друг другу; «c» и «d» необязательны. Но для «b» требуются «c» и «d», поэтому, если используется «b», также должны быть определены «c» и «d». Значение по умолчанию не определено; т.е. это не относится к большинству атрибутов;

– «capability-reference» - (string) - если значение определено, это означает, что значение этого атрибута определяет динамическую часть названия указанной возможности, предоставляемой другим ресурсом. Это означает, что атрибут является ссылкой на другую область модели управления (обратите внимание, что в настоящее время некоторые атрибуты, которые ссылаются на другие области модели, могут не предоставлять эту информацию);

– «head-comment-allowed» - (boolean) - этот ключ описания предназначен для возможного использования в будущем;

– «tail-comment-allowed» - (boolean) - этот ключ описания предназначен для возможного использования в будущем;

– произвольные пары «ключ/значение», которые дополнительно описывают значение атрибута, например «max ⇒ 2». Смотрите раздел «Произвольные дескрипторы».

Вот несколько примеров:

```
"foo" => {
   "description" => "The foo",
   "type" => INT,
   "max" => 2
}
```

```
"bar" => {
  "description" => "The bar",
  "type" => OBJECT,
  "value-type" => {
     "size" => INT,
     "color" => STRING
  }
}
```

*Описание операции*.

С ресурсом управления могут быть связаны операции. Описание операции будет включать следующую информацию:

– «operation-name» - (String) - название операции;

– «description» - (String) - текстовое описание операции;

– «request-properties» - (преобразование «String» в сложную структуру) -;

– «reply-properties» - (сложная структура или пустая) - описание возвращаемого значения операции с пустым узлом, означающим «void». Смотрите ниже подробное описание типов возвращаемых значений операций;

– «restart-required» - (String) - один из вариантов «no-services», «all-services», «resource-services» или «jvm». Указывает, вносит ли операция изменение конфигурации, требующее перезапуска служб (или всей JVM), чтобы изменение вступило в силу во время выполнения. Смотрите обсуждение применения обновлений к службам среды выполнения ниже. Значение по умолчанию – «no-services».

Описание операционного параметра или возвращаемого значения:

– «description» - (String) - текстовое описание параметра или возвращаемого значения;

– «type» - (org.jboss.dmr.ModelType) - тип параметра или возвращаемого значения. Одно из значений перечисления: «BIG_DECIMAL», «BIG_INTEGER», «BOOLEAN», «BYTES», «DOUBLE», «INT», «LIST», «LONG», «OBJECT», «PROPERTY», «STRING»;

– «value-type» - (ModelType или сложная структура) - присутствует только в том случае, если тип «LIST» или «OBJECT». Если все элементы в «LIST» или все значения типа «OBJECT» относятся к одному и тому же типу, то это будет одно из перечислений типа «ModelType»: «BIG_DECIMAL», «BIG_INTEGER», «BOOLEAN», «BYTES», «DOUBLE», «INT», «LIST», «LONG», «PROPERTY», «STRING». В противном случае параметр «value-type» будет содержать подробную информацию о структуре атрибута «value». Итак, параметр с типом «LIST» и значением типа «ModelType.STRING» аналогичен Java «List<String>», а параметр со значением типа «Modeltype.INT» аналогичен Java «List<Integer>». Параметр с типом «OBJECT» и значением типа «ModelType.STRING» аналогична Java «Map<String, String>». Параметр с типом «OBJECT» и «value-type», значение которого не относится к типу «ModelType» представляет собой полностью определенный сложный объект с описанными правовыми полями объекта и их значениями;

– «expressions-allowed» - (boolean) - указывает, может ли значение параметра или возвращаемое значение быть типа «ModelType.EXPRESSION», а не его стандартный тип (см. Тип и тип значения выше для обсуждения стандартного типа.) Значение «ModelType.EXPRESSION» содержит выражение для подстановки системного свойства или переменной среды, которое сервер разрешит с помощью сопоставления системных свойств на стороне сервера перед использованием значения. Например, параметр с именем «max-threads» может иметь значение выражения «${example.pool.max-threads:10}» вместо всего «10». Значение по умолчанию, если его нет, равно «false». Более подробное описание смотрите в подразделе «Разрешение выражений»;

– «required» - (boolean) - значение «true», если параметр или возвращаемое значение должны иметь определенное значение в операции или ответе, если не определен другой элемент, включенный в список альтернатив; значение «false», если оно может быть

неопределенным (подразумевая нулевое значение) даже при отсутствии альтернатив. Если нет, значение «true» используется по умолчанию;

– «nillable» - (boolean) - значение равно «true», если параметр или возвращаемое значение могут не иметь определенного значения в представлении своей части модели. Параметр или возвращаемое значение с возможностью обнуления могут быть не определены либо потому, что они не требуются, либо потому, что они требуются, но имеют альтернативы, и одна из альтернатив определена;

– «default» - значение по умолчанию для параметра, который будет использоваться в службах «runtime services», если этот параметр явно не определен и другие параметры, перечисленные в качестве альтернативных, не определены;

– «restart-required» - (String) - Один из «no-services», «all-services», «resource-services» или «jvm». Относится только к атрибутам, тип доступа к которым «read-write». Указывает, требуется ли для выполнения операции «write-attribute», в параметре «name» которой указан этот атрибут, перезапуск служб (или всей виртуальной машины JVM), чтобы изменения вступили в силу во время выполнения. Смотрите обсуждение вопроса Применение обновлений к службам среды выполнения ниже. Значение по умолчанию «no-services»;

– «alternatives» - (Список строк) - указывает на исключительную связь между параметрами. Если этот атрибут определен, другие параметры, перечисленные в значении этого дескриптора, должны быть не определены, даже если их обязательный дескриптор указывает значение «true»; т.е. наличие этого параметра удовлетворяет требованию. Обратите внимание, что параметр, который явно не настроен, но имеет значение по умолчанию, по-прежнему считается не определенным для проверки того, не нарушена ли исключительная связь. Значение по умолчанию не определено, т.е. это не относится к большинству параметров;

– «requires» - (Список строк) - указывает, что если этот параметр имеет значение (отличное от «undefined»), то другие параметры, указанные в значении этого дескриптора, также должны иметь значение, даже если их обязательный дескриптор указывает значение «false». Обычно это используется в сочетании с альтернативными вариантами. Например, параметры «a» и «b» обязательны, но являются альтернативами друг другу; «c» и «d» необязательны. Но для «b» требуются «c» и «d», поэтому, если используется «b», также должны быть определены «c» и «d». Значение по умолчанию не определено, т.е. это не относится к большинству параметров;

– произвольные пары ключ/значение, которые дополнительно описывают значение атрибута, например «max ⇒2». Смотрите раздел Произвольные дескрипторы.

*Произвольные дескрипторы*.

Описание атрибута, параметра операции или типа возвращаемого значения операции может включать произвольные пары ключ/значение, которые предоставляют дополнительную информацию. Наличие конкретной пары ключ/значение зависит от контекста, например, пара с ключом «max», вероятно, встречается только как часть описания некоторого числового типа.

Ниже приведены стандартные ключи и тип их ожидаемого значения. Если авторы дескриптора хотят добавить произвольную пару ключ/значение к некоторому дескриптору и семантика соответствует значению одного из следующих элементов, необходимо использовать стандартный тип ключ/значение.

– «min» - (int) - минимальное значение некоторого числового типа. Отсутствие этого элемента означает, что минимального значения не существует.

– «max» - (int) - максимальное значение некоторого числового типа. Отсутствие этого элемента означает, что максимального значения не существует.

– «min-length» - (int) - минимальная длина некоторого типа строки, списка или байта[]. Отсутствие этого элемента означает, что минимальная длина равна нулю.

– «max-length» - (int) - максимальная длина некоторой строки, списка или байта[]. Отсутствие этого элемента означает, что не существует максимального значения.

– «allowed» - (List) - список допустимых значений. Тип элементов в списке должен соответствовать типу атрибута.

– «unit» - (Единица измерения стоимости, если таковая применима) - например, «ns», «ms», «s», «m», «h», КБ, МБ, ТБ. Посмотрите на «org.jboss.as.controller.client.helpers.MeasurementUnit» в «org.jboss.as:jboss-as-controllerclient» артефакт для составления списка разрешенных единиц измерения.

– «filesystem-path» - (boolean) - флаг, указывающий на то, что атрибут является путем к файловой системе.

– «attached-streams» - (boolean) - флаг, указывающий на то, что атрибут является идентификатором потока для присоединенного потока.

– «relative-to» - (boolean) - флаг, указывающий на то, что атрибут является относительным путем.

– «feature-reference» - (boolean) - флаг, указывающий на то, что атрибут является ссылкой на функцию подготовки с помощью возможности.

Вот несколько примеров:

```
{
   "operation-name" => "incrementFoo",
   "description" => "Increase the value of the 'foo' attribute by the
   given amount",
   "request-properties" => {
      "increment" => {
        "type" => INT,
        "description" => "The amount to increment",
        "required" => true
   }},
   "reply-properties" => {
        "type" => INT,
        "description" => "The new value",
   }
}
```

```
{
 "operation-name" => "start",
 "description" => "Starts the thing",
 "request-properties" => {},
 "reply-properties" => {}
}
```

*Описание отношений между родительскими и дочерними узлами*.

Адрес, используемый для адресации адресуемой части модели, должен представлять собой упорядоченный список пар «ключ-значение». Результатом этого требования является то, что адресуемые части модели естественным образом образуют древовидную структуру, родительские узлы которой определяют допустимые ключи, а дочерние узлы - допустимые значения. Родительский узел также определяет мощность связи. Описание родительского узла включает в себя дочерний элемент, который описывает эти взаимосвязи:

```
{
   ....
   "children" => {
      "connector" => {
          .... description of the relationship with children of type
          "connector"
      },
      "virtual-host" => {
          .... description of the relationship with children of type
          "virtualhost"
```

```
      }
}
```

Описание каждой взаимосвязи будет включать следующие элементы:

– «description» - (String) - текстовое описание взаимосвязи;

– «model-description» - (либо «undefined», либо сложная структура) - это узел типа «ModelType.OBJECT», ключи которого являются допустимыми значениями для части значений адреса ресурса этого типа, со специальным символом «*», указывающим, что часть значений может иметь произвольное значение. Значения в узле представляют собой полное описание конкретного дочернего ресурса (его текстовое описание, атрибуты, операции, дочерние элементы), как описано выше. Это описание модели также может быть «undefined», т.е. иметь нулевое значение, если запрос, запрашивающий описание родительского узла, не включал «recursive» параметр, имеющий значение «true».

Пример с тем, был ли флаг рекурсии установлен в значение «true»:

```
{
   "description" => "The connectors used to handle client
   connections",
   "model-description" => {
     "*" => {
        "description" => "Handles client connections",
        "min-occurs" => 1,
        "attributes => {
           ... details of children as documented above
        },
        "operations" => {
           .... details of operations as documented above
        },
        "children" => {
           .... details of the children's children
        }
     }
   }
}
```

Если рекурсивный флаг был ложным:

```
{
  "description" => "The connectors used to handle client connections",
  "model-description" => undefined
}
```

*Применение обновлений к службам среды выполнения*

Описание атрибута или операции может содержать дескриптор, необходимый для перезапуска; в этом разделе объясняется значение этого дескриптора.

Операция, которая изменяет постоянную конфигурацию ресурса управления, обычно также может повлиять на службу времени выполнения, связанную с этим ресурсом. Например, существует служба среды выполнения, связанная с любым элементом «host.xml» или «standalone.xml <интерфейс>»; другие службы в среде выполнения зависят от этой службы, предоставляя «InetAddress», связанный с интерфейсом. Во многих случаях обновление постоянной конфигурации ресурса может быть немедленно применено к соответствующей службе времени выполнения. Состояние службы времени выполнения обновляется, чтобы отразить новые значения.

Однако во многих случаях состояние службы времени выполнения не может быть обновлено без перезапуска службы. Перезапуск службы может иметь серьезные последствия. Перезапуск службы «A» вызовет перезапуск других служб «B», «C» и «D», зависящих от «A», что приведет к перезапуску служб, зависящих от «B», «C» и «D» и т.д. Эти перезапуски служб могут привести к нарушению обработки запросов конечных пользователей.

Поскольку перезапуск службы может нарушить обработку запросов конечного пользователя, обработчики операций управления не будут перезапускать какую-либо службу без какой-либо явной инструкции от конечного пользователя, указывающей на желательность перезапуска службы. В некоторых случаях простое выполнение операции указывает на то, что пользователь хочет перезапустить службы (например, «/host=master/server-config=serverone:restart» операция в управляемом домене или «/:reload» работа на автономном сервере). Во всех остальных случаях, если операция (или запись атрибута) не может быть выполнена без перезапуска служба, метаданные, описывающие операцию или атрибут, будут включать дескриптор, необходимый для «restart-required», значение которого указывает, что необходимо для того, чтобы операция повлияла на время выполнения:

– «no-services» - применение операции к среде выполнения не требует перезапуска каких-либо служб. Это значение используется по умолчанию, если дескриптор, требующий перезапуска, отсутствует;

– «all-services» - операция может только немедленно обновить постоянную конфигурацию; применение операции к среде выполнения потребует последующего перезапуска всех служб в уязвимой виртуальной машине. Выполнение операции переведет сервер в состояние, требующее перезагрузки. До тех пор, пока не будет выполнен перезапуск всех служб, ответ на эту операцию и на любую последующую операцию будет содержать заголовок ответа «process-state» ⇒ «reload-required». Для автономного сервера перезапуск всех служб можно выполнить, выполнив команду «reload» CLIP. Для сервера в управляемом домене перезапуск всех служб выполняется с помощью операции перезагрузки, нацеленной на конкретный сервер (например, «/host=master/server=server-one:reload»);

– «jvm» - операция может только немедленно обновить постоянную конфигурацию; применение операции к среде выполнения потребует полного перезапуска процесса (т.е. остановки JVM и запуска новой JVM). Выполнение операции переведет сервер в состояние, требующее перезапуска. До тех пор, пока не будет выполнен перезапуск, ответ на эту операцию и на любую последующую операцию будет содержать заголовок ответа «process-state» ⇒ «restart-required». Для автономного сервера полный перезапуск процесса требует предварительной остановки сервера с помощью операций на уровне операционной системы (Ctrl-C, kill) или с помощью выключения Команду CLI, а затем снова запустите сервер из командной строки. Для сервера в управляемом домене перезапуск сервера требует выполнения операции «/host=<host>/serverconfig=<server>:restart»;

– «resource-services» - операция может только немедленно обновить постоянную конфигурацию; применение операции к среде выполнения потребует последующего перезапуска некоторых служб, связанных с ресурсом. Если операция содержит заголовок запроса «allow-resourceservice-restart» ⇒ «true», обработчик операции продолжит выполнение и перезапустит службу времени выполнения. В противном случае выполнение операции переведет сервер в состояние, требующее перезагрузки (смотрите обсуждение всех служб выше, чтобы узнать больше о состоянии, требующем перезагрузки).

### 10.3.1 Разрешение выражения

При разрешении выражения в модели проверяются следующие местоположения. Для этого примера мы будем использовать выражение «${my.example-expr}».

– Сначала мы проверяем, существует ли системное свойство с именем «my.example-expr». Если оно есть, мы используем его значение в качестве результата разрешения. Если нет, мы продолжаем проверку следующих местоположений.

– Мы преобразуем имя «my.example-expr» в верхний регистр и заменяем все не буквенно-цифровые символы на символы подчеркивания, в результате чего получаем «MY_EXAMPLE_EXPR». Мы проверяем, существует ли переменная окружения с таким именем. Если есть, мы используем его значение в качестве результата разрешения. Если нет, мы продолжаем проверку следующего местоположения.

**Внимание:** этот шаг был введен для WildBoss Pro 25 и позволяет в особых случаях создавать некоторые проблемы. Допустим, у вас уже есть переменная окружения «COMMON_VAR_NAME=foo», и вы используете «${common-var-name:bar}» в конфигурации WildBoss Pro. До версии WildBoss Pro 25 будет использоваться значение по умолчанию (например, «bar»). В WildBoss Pro 25 и более поздних версиях будет использоваться значение из переменной окружения (например, «foo»).

– Если (и только если) исходное имя начинается с «env.», мы удаляем префикс и ищем переменную окружения с именем «то, что у нас осталось», без выполнения преобразования (например, если исходное имя было «env.example», мы ищем переменную окружения с именем «example»; если исходное имя было envy)..MY_EXAMPLE_EXPR, мы ищем переменную окружения с именем «MY_EXAMPLE_EXPR»). Если такая переменная окружения существует, мы используем ее значение в качестве результата разрешения.

– Если ни одна из вышеперечисленных проверок не дала результата, разрешение завершилось ошибкой. Последний шаг - проверить, задано ли в выражении значение по умолчанию. В нашем примере с «${my.example-expr}» значение по умолчанию не задано, поэтому выражение не удалось разрешить. Если мы указали значение по умолчанию в выражении, то возвращается значение по умолчанию (например, для «${my.example-expr:hello}» возвращается значение «hello»).

## 10.4 API управления HTTP

### 10.4.1 Вступление

API управления в WildBoss Pro доступен по нескольким каналам, одним из которых является HTTP и JASON.

Даже если вы еще не использовали командную строку «curl», возможно, уже использовали этот канал, поскольку именно так веб-консоль взаимодействует с API управления.

По умолчанию WildBoss Pro распространяется с защитой, механизм защиты по умолчанию основан на имени пользователя и пароле и использует HTTP-дайджест для процесса аутентификации.

Таким образом, вам необходимо создать пользователя с помощью скрипта «add-user.sh».

### 10.4.2 Взаимодействие с моделью

Поскольку мы должны пройти аутентификацию, клиент должен будет поддерживать дайджест-аутентификацию HTTP.

Например, это можно активировать в «curl» с помощью опции «--digest».

API управления HTTP WildBoss Pro придерживается принципов REST, поэтому операции GET должны быть идемпотентными.

Это означает, что для чтения модели можно использовать запрос с методом GET, но вы не сможете его изменить.

Для изменения модели или ее чтения необходимо использовать POST. Запрос POST может содержать операцию либо в формате DMR, либо в формате JSON в качестве тела.

Вы должны указать заголовок Content-Type=application/json в запросе, чтобы указать, что вы используете какой-либо JSON.

Если вы хотите отправить DMR в теле запроса, то заголовок «Content-Type» или «Accept» должен быть «application/dmr-encoded».

### 10.4.3 GET для прочтения

Хотя с помощью POST вы можете делать все, что угодно, некоторые операции могут быть вызваны с помощью «классического» запроса GET.

Это поддерживаемые операции для GET:

– «attribute»: для операции чтения атрибута;

– «resource»: для операции чтения ресурсов;

– «resource-description»: для операции чтения описания ресурса;

– «snapshots»: для операции создания моментальных снимков списка;

– «operation-description»: для операции чтения описания операции;

– «operation-names»: для операции чтения объявлений с именами.

Формат URL-адреса следующий: *http://server:9990/management/*
*<path_to_resource>?operation=<operation_name>&operation_parameter=<value>...*

*path_to_resource* is the path to the wanted resource replacing all '=' with '/' : thus for example subsystem=undertow/server=default-server becomes subsystem/undertow/server/default-server.

Итак, чтобы прочитать состояние сервера:

```
http://localhost:9990/management?operation=attribute&name=server-
state&json.pretty=1
```

### 10.4.4 Давайте почитаем какой-нибудь ресурс

1) Это простая операция, эквивалентная запуску «:read-attribute(name=server-state)» с помощью CLI в корневом каталоге:

– использование GET;

```
http://localhost:9990/management?operation=attribute&name=serverstate&
json.pretty=1
```

– использование POST.

```
$ curl --digest -L -D - http://localhost:9990/management --header
"Content-Type:
application/json" -d '{"operation":"read-
attribute","name":"serverstate","
json.pretty":1}' -u admin
Enter host password for user 'admin':
HTTP/1.1 401 Unauthorized
Connection: keep-alive
WWW-Authenticate: Digest
realm="ManagementRealm",domain="/management",nonce="P80WU3BANtQNMTQwNjg
5Mzc5MDQ2
MlpjmRaZ+Vlp1OVeNEGBeXg=",opaque="00000000000000000000000000000000",alg
orithm=MD
5
Content-Length: 77
Content-Type: text/html
Date: Fri, 01 Aug 2014 11:49:50 GMT
HTTP/1.1 200 OK
Connection: keep-alive
Authentication-Info:
nextnonce="M+h9aADejeINMTQwNjg5Mzc5MDQ2OPQbHKdAS8pRE8BbGEDY5uI="
Content-Type: application/json; charset=utf-8
```

```
Content-Length: 55
Date: Fri, 01 Aug 2014 11:49:50 GMT
{
 "outcome" : "success",
 "result" : "running"
}
```

2) Вот пример операции с ресурсом с вложенным адресом и переданными параметрами. Это то же самое, что если бы вы запустили «/host=master/server=server-01:readattribute(name=server-state)».

```
$ curl --digest -L -D - http://localhost:9990/management --header
"Content-Type:
application/json" -d '{"operation":"readattribute","
address":[{"host":"master"},{"server":"server-
01"}],"name":"serverstate","
json.pretty":1}'
HTTP/1.1 200 OK
Transfer-encoding: chunked
Content-type: application/json
Date: Tue, 17 Apr 2012 04:02:24 GMT
{
 "outcome" : "success",
 "result" : "running"
}
```

3) В следующем примере мы получим информацию из http-соединения в подсистеме «undertow», включая атрибуты времени выполнения Это то же самое, что запуск «/subsystem=undertow/server=default-server:read-resource(includeruntime=true,recursive=true)» в CLI:

– использование GET;

```
http://localhost:9990/management/subsystem/undertow/server/defaultserve
r?operation=resource&recursive=true&json.pretty=1
{
   "default-host" : "default-host",
   "servlet-container" : "default",
   "ajp-listener" : null,
   "host" : {"default-host" : {
     "alias" : ["localhost"],
     "default-web-module" : "ROOT.war",
     "filter-ref" : {
       "server-header" : {"predicate" : null},
       "x-powered-by-header" : {"predicate" : null}
     },
     "location" : {"/" : {
       "handler" : "welcome-content",
       "filter-ref" : null
     }},
     "setting" : null
   }},
   "http-listener" : {"default" : {
     "allow-encoded-slash" : false,
     "allow-equals-in-cookie-value" : false,
     "always-set-keep-alive" : true,
     "buffer-pipelined-data" : true,
```

```
    "buffer-pool" : "default",
    "certificate-forwarding" : false,
    "decode-url" : true,
    "enabled" : true,
    "max-buffered-request-size" : 16384,
    "max-cookies" : 200,
    "max-header-size" : 51200,
    "max-headers" : 200,
    "max-parameters" : 1000,
    "max-post-size" : 10485760,
    "proxy-address-forwarding" : false,
    "read-timeout" : null,
    "receive-buffer" : null,
    "record-request-start-time" : false,
    "redirect-socket" : "https",
    "send-buffer" : null,
    "socket-binding" : "http",
    "tcp-backlog" : null,
    "tcp-keep-alive" : null,
    "url-charset" : "UTF-8",
    "worker" : "default",
    "write-timeout" : null
  }},
  "https-listener" : null
}
```

– использование POST.

```
$ curl --digest -D - http://localhost:9990/management --header
"Content-Type:
application/json" -d '{"operation":"read-resource", "include-
runtime":"true" ,
"recursive":"true",
"address":["subsystem","undertow","server","defaultserver"],
"json.pretty":1}' -u admin:admin
HTTP/1.1 401 Unauthorized
Connection: keep-alive
WWW-Authenticate: Digest
realm="ManagementRealm",domain="/management",nonce="a3paQ9E0/l8NMTQwNjg
5OTU0NDk4
OKjmim2lopZNc5zCevjYWpk=",opaque="00000000000000000000000000000000",alg
orithm=MD
5
Content-Length: 77
Content-Type: text/html
Date: Fri, 01 Aug 2014 13:25:44 GMT
HTTP/1.1 200 OK
Connection: keep-alive
Authentication-Info:
nextnonce="nTOSJd3ufO4NMTQwNjg5OTU0NDk5MeUsRw5rKXUT4Qvk1nbrG5c="
Content-Type: application/json; charset=utf-8
Content-Length: 1729
Date: Fri, 01 Aug 2014 13:25:45 GMT
{
 "outcome" : "success",
 "result" : {
   "default-host" : "default-host",
```

```
    "servlet-container" : "default",
    "ajp-listener" : null,
    "host" : {"default-host" : {
      "alias" : ["localhost"],
      "default-web-module" : "ROOT.war",
      "filter-ref" : {
        "server-header" : {"predicate" : null},
        "x-powered-by-header" : {"predicate" : null}
      },
      "location" : {"/" : {
        "handler" : "welcome-content",
        "filter-ref" : null
      }},
      "setting" : null
    }},
    "http-listener" : {"default" : {
      "allow-encoded-slash" : false,
      "allow-equals-in-cookie-value" : false,
      "always-set-keep-alive" : true,
      "buffer-pipelined-data" : true,
      "buffer-pool" : "default",
      "certificate-forwarding" : false,
      "decode-url" : true,
      "enabled" : true,
      "max-buffered-request-size" : 16384,
      "max-cookies" : 200,
      "max-header-size" : 51200,
      "max-headers" : 200,
      "max-parameters" : 1000,
      "max-post-size" : 10485760,
      "proxy-address-forwarding" : false,
      "read-timeout" : null,
      "receive-buffer" : null,
      "record-request-start-time" : false,
      "redirect-socket" : "https",
      "send-buffer" : null,
      "socket-binding" : "http",
      "tcp-backlog" : null,
      "tcp-keep-alive" : null,
      "url-charset" : "UTF-8",
      "worker" : "default",
      "write-timeout" : null
    }},
    "https-listener" : null
  }
}
```

4) Вы также можете использовать какой-нибудь закодированный DMR, но результат не будет удобочитаемым для человека.

```
curl --digest -u admin:admin --header "Content-Type: application/dmr-
encoded" -
dbwAAAAMACW9wZXJhdGlvbnMADXJlYWQtcmVzb3VyY2UAB2FkZHJlc3NsAAAAAAAHcmVjdX
JzZVoB
http://localhost:9990/management
```

5) Вы можете развертывать приложения на сервере:

– сначала загрузите файл, который создаст управляемый контент. Вам нужно будет использовать «http://localhost:9990/management/add-content»;

```
curl --digest -u admin:admin --form file=@tiny-webapp.war
http://localhost:9990/management/add-content
{"outcome" : "success", "result" : { "BYTES_VALUE" :
"+QJlHTDrogO9pm/57GkT/vxWNz0=" }}
```

– теперь давайте развернем приложение.

```
curl --digest -u admin:admin -L --header "Content-Type:
application/json" -d
'{"content":[{"hash": {"BYTES_VALUE" :
"+QJlHTDrogO9pm/57GkT/vxWNz0="}}],
"address": [{"deployment":"tiny-webapp.war"}], "operation":"add",
"enabled":"true"}' http://localhost:9990/management
{"outcome" : "success"}
```

### 10.4.5 Использование некоторого кода веб-служб Jakarta RESTful Web Services

```
HttpAuthenticationFeature feature =
HttpAuthenticationFeature.digest("admin", "admin"
);
Client client = ClientBuilder.newClient();
client.register(feature);
Entity<SimpleOperation> operation = Entity.entity(
new SimpleOperation("read-resource", true, "subsystem", "undertow",
"server",
"default-server"),
MediaType.APPLICATION_JSON_TYPE);
WebTarget managementResource =
client.target("http://localhost:9990/management");
String response =
managementResource.request(MediaType.APPLICATION_JSON_TYPE)
.header("Content-type", MediaType.APPLICATION_JSON)
.post(operation, String.class);
System.out.println(response);
{"outcome" : "success", "result" : {"default-host" : "default-host",
"servletcontainer"
: "default", "ajp-listener" : null, "host" : {"default-host" :
{"alias" :
["localhost"], "default-web-module" : "ROOT.war", "filter-ref" :
{"server-header" : {
"predicate" : null}, "x-powered-by-header" : {"predicate" : null}},
"location" : {"/"
: {"handler" : "welcome-content", "filter-ref" : null}}, "setting" :
null}}, "httplistener"
: {"default" : {"allow-encoded-slash" : false, "allow-equals-in-
cookievalue"
: false, "always-set-keep-alive" : true, "buffer-pipelined-data" :
true,
"buffer-pool" : "default", "certificate-forwarding" : false, "decode-
url" : true,
"enabled" : true, "max-buffered-request-size" : 16384, "max-cookies" :
200, "maxheader-
```

```
size" : 51200, "max-headers" : 200, "max-parameters" : 1000, "max-
post-size" :
10485760, "proxy-address-forwarding" : false, "read-timeout" : null,
"receive-buffer"
: null, "record-request-start-time" : false, "redirect-socket" :
"https", "sendbuffer"
: null, "socket-binding" : "http", "tcp-backlog" : null, "tcp-keep-
alive" :
null, "url-charset" : "UTF-8", "worker" : "default", "write-timeout" :
null}}, "httpslistener"
: null}}
```

### 10.5 Собственный API управления

Автономный процесс WildBoss Pro или управляемый доменный контроллер домена или подчиненный хост-контроллер могут быть сконфигурированы для прослушивания запросов на удаленное управление с помощью своего «собственного интерфейса управления»:

```
<native-interface interface="management" port="9999" security-
realm="ManagementRealm"/>
```

(Смотри «standalone/configuration/standalone.xml or domain/configuration/host.xml»)

Инструмент CLI, поставляемый с сервером приложений, использует этот интерфейс, и пользователь может разрабатывать пользовательские клиенты, которые также используют его. В этом разделе мы рассмотрим основы разработки такого клиента. Мы также подробно рассмотрим формат запросов и ответов на операции низкоуровневого управления - информация, которая также должна оказаться полезной для пользователей инструмента CLI.

#### 10.5.1 Собственное управление клиентскими зависимостями

Собственный интерфейс управления использует открытый протокол, основанный на библиотеке удаленного взаимодействия JBoss. JBoss удаленное взаимодействие используется для установления канала связи между клиентом и управляемым процессом. Как только канал связи установлен, основным трафиком по каналу являются запросы на управление, инициируемые клиентом, и асинхронные ответы от целевого процесса.

Пользовательский клиент на базе Java должен иметь артефакт «maven» «org.jboss.as:jboss-as-controllerclient» и его зависимости от пути к классу. Другими зависимостями являются (см. таблицу 34).

Таблица 34

| Артефакт «Maven» | Цель |
|---|---|
| org.jboss.remoting:jboss-remoting | удаленная связь |
| org.jboss:jboss-dmr | подробное представление модели управления |
| org.jboss.as:jboss-as-protocol | проводной протокол для удаленного управления WildBoss Pro |
| org.jboss.sasl:jboss-sasl | аутентификация SASL |
| org.jboss.xnio:xnio-api | неблокирующий ввод-вывод |
| org.jboss.xnio:xnio-nio | неблокирующий ввод-вывод |
| org.jboss.logging:jboss-logging | регистрация |
| org.jboss.threads:jboss-threads | управление потоками |

| | |
|---|---|
| org.jboss.marshalling:jboss-marshalling | сортировка и отмена сортировки данных в потоках и из них |

Клиентский API полностью находится в артефакте «org.jboss.as:jboss-as-controller-client»; остальные зависимости являются частью внутренней реализации «org.jboss.as:jboss-as-controller-client» и не зависят от времени компиляции какого-либо пользовательского клиента, основанного на нем.

Протокол управления является открытым протоколом, поэтому полностью настраиваемый клиент может быть разработан без использования этих библиотек (например, с использованием Python или какого-либо другого языка).

### 10.5.2 Работа с ModelControllerClient

Класс «org.jboss.as.controller.client.ModelControllerClient» - это основной класс, который пользовательский клиент будет использовать для управления экземпляром сервера WildBoss Pro, контроллером домена или подчиненным хост-контроллером.

Пользовательский клиент должен иметь артефакт «maven» «org.jboss.as:jboss-as-controller-client» и его зависимости от пути к классу.

*Создание ModelControllerClient*.

Чтобы создать клиент управления, который может подключаться к встроенному сокету управления вашего целевого процесса, просто:

```
ModelControllerClient client =
ModelControllerClient.Factory.create(InetAddress.getByName("localhost")
, 9999);
```

Адрес и порт - это то, что настроено в элементе <management><managementinterfaces><native-interface…/>.

Однако, как правило, собственный интерфейс управления защищен и требует от клиентов аутентификации. На стороне клиента пользовательскому клиенту необходимо будет предоставить учетные данные для аутентификации пользователя, полученные любым удобным для клиента способом (например, из диалогового окна в Клиент на основе графического интерфейса пользователя.) Доступ к этим учетным данным предоставляется путем передачи реализации «javax.security.auth.callback.CallbackHandler». Например:

```
static ModelControllerClient createClient(final InetAddress host,
       final int port,final String username, final char[] password,
       final String
securityRealmName) {
  final CallbackHandler callbackHandler = new CallbackHandler() {
    public void handle(Callback[] callbacks) throws IOException,
UnsupportedCallbackException {
 for (Callback current : callbacks) {
    if (current instanceof NameCallback) {
       NameCallback ncb = (NameCallback) current;
       ncb.setName(username);
    } else if (current instanceof PasswordCallback) {
        PasswordCallback pcb = (PasswordCallback) current;
        pcb.setPassword(password.toCharArray());
    } else if (current instanceof RealmCallback) {
        RealmCallback rcb = (RealmCallback) current;
        rcb.setText(rcb.getDefaultText());
    } else {
       throw new UnsupportedCallbackException(current);
    }
```

```
    }
   }
  };
return ModelControllerClient.Factory.create(host, port,
callbackHandler);
}
```

*Создание объекта запроса на операцию*

Запросы на управление формулируются с использованием класса «org.jboss.dmr.ModelNode» из библиотеки «jboss-dmr» . Библиотека «jboss-dmr» позволяет выразить полную модель управления WildBoss Pro с помощью очень небольшого числа Java-типов. Более подробную информацию об использовании этой библиотеки смотрите в разделе «Подробное управление» и в библиотеке «jboss-dmr».

Давайте покажем пример создания объекта запроса операции, который можно использовать для чтения описания ресурса HTTP-коннектора веб-подсистемы:

```
ModelNode op = new ModelNode();
op.get("operation").set("read-resource-description");
ModelNode address = op.get("address");
address.add("subsystem", "web");
address.add("connector", "http");
op.get("recursive").set(true);
op.get("operations").set(true);
```

Что мы здесь сделали, так это создали ModelNode типа «ModelType.OBJECT» со следующими полями:

– «operation» - имя вызываемой операции. Все запросы на выполнение операции должны содержать это поле, а его значение должно быть строкой;

– «address» - адрес ресурса, для которого выполняется операция. Это поле должно иметь тип «ModelType.LIST», каждый элемент которого относится к типу «ModelType.PROPERTY». Если это поле опущено, операция будет направлена на корневой ресурс. Операция может быть нацелена на любой адрес в модели управления; здесь мы нацеливаем ее на ресурс http -соединителя веб-подсистемы.

В этом случае запрос включает в себя два необязательных параметра:

– «recursive» - означает, что вы хотите получить описание дочерних ресурсов под этим ресурсом. Значение по умолчанию равно «false»;

– «operations» - значение «true» означает, что вы хотите включить описание операций, предоставляемых ресурсом. Значение по умолчанию равно «false».

Разные операции принимают разные параметры, а некоторые вообще не принимают никаких параметров.

Более подробную информацию о структуре узла модели, который будет представлять запрос на операцию, смотрите в разделе Формат запроса на операцию с определенным типом.

В приведенном выше примере создается узел модели запроса на операцию, эквивалентный тому, который создается внутри CLI при анализе и выполнении следующей низкоуровневой команды CLI:

```
[localhost:9999 /] /subsystem=web/connector=http:read-
resourcedescription(
recursive=true,operations=true)
```

Выполните операцию и манипулируйте результатом:

Метод «execute» отправляет узел модели запроса операции управляемому процессу и возвращает узел модели, содержащий ответ процесса:

```
ModelNode returnVal = client.execute(op);
System.out.println(returnVal.get("result").toString());
```

Общие сведения о структуре «returnVal» смотри в разделе «Формат ответа на типизированную операцию ModelNode».

Операция «execute», показанная выше, блокирует вызывающий поток до тех пор, пока не будет получен ответ от управляемого процесса. ModelControllerClient также предоставляет API, позволяющий выполнять асинхронный вызов:

```
Future<ModelNode> future = client.executeAsync(op);
. . . // do other stuff
ModelNode returnVal = future.get();
System.out.println(returnVal.get("result").toString());
```

Закройте ModelControllerClient.

ModelControllerClient можно повторно использовать для нескольких запросов. Создание нового ModelControllerClient для каждого запроса является нарушением шаблона. Однако, когда ModelControllerClient больше не нужен, он всегда должен быть явно закрыт, что позволяет ему закрывать любые подключения к процессу, которым он управлял, и освобождать другие ресурсы:

```
client.close();
```

### 10.5.3 Формат детализированного запроса на выполнение операции

Базовый метод, который мог бы использовать пользователь WildBoss Pro 26.1 программного управления API, очень прост:

```
ModelNode execute(ModelNode operation) throws IOException;
```

где возвращаемое значение — это детализированное представление ответа, а «operation» — это детализированное представление вызываемой операции.

Цель этого раздела - задокументировать структуру работы.

Смотрите раздел «Формат ответа на типизированную операцию для обсуждения формата ответа».

Простые операции.

Текстовое представление простой операции выглядело бы следующим образом:

```
{
  "operation" => "write-attribute",
  "address" => [
     ("profile" => "production"),
     ("subsystem" => "threads"),
     ("bounded-queue-thread-pool" => "pool1")
  ],
  "name" => "count",
  "value" => 20
}
```

Java-код для получения этого результата был бы следующим:

```
ModelNode op = new ModelNode();
op.get("operation").set("write-attribute");
ModelNode addr = op.get("address");
addr.add("profile", "production");
addr.add("subsystem", "threads");
addr.add("bounded-queue-thread-pool", "pool1");
op.get("name").set("count");
op.get("value").set(20);
System.out.println(op);
```

Порядок, в котором самые удаленные элементы отображаются в запросе, не имеет значения. Обязательными элементами являются:

– «operation» - (String) - название вызываемой операции;

– «address» - адрес управляемого ресурса, к которому должен быть выполнен запрос. Если этот параметр не задан, то это адрес корневого ресурса. Адрес представляет собой упорядоченный список пар ключ-значение, описывающий местоположение ресурса в общем дереве ресурсов управления. Ресурсы управления организованы в виде дерева, поэтому важен порядок расположения элементов в адресе.

Другими парами ключ/значение являются имена параметров и их значения. Имена и значения должны соответствовать тому, что указано в описании операции.

Параметры могут иметь любое имя, за исключением зарезервированных слов операция, адрес и заголовки операций.

Заголовки операций.

Помимо специальных значений операций и адресов, рассмотренных выше, запросы на выполнение операций могут также содержать специальные значения «заголовков», которые помогают управлять выполнением операции. Эти заголовки создаются в соответствии со специальным зарезервированным словом операция-заголовки:

```
ModelNode op = new ModelNode();
op.get("operation").set("write-attribute");
ModelNode addr = op.get("address");
addr.add("profile", "production");
addr.add("subsystem", "threads");
addr.add("bounded-queue-thread-pool", "pool1");
op.get("name").set("count");
op.get("value").set(20);
System.out.println(op);
```

Это приводит к:

```
{
 "operation" => "write-attribute",
 "address" => [
    ("profile" => "production"),
    ("subsystem" => "threads"),
    ("bounded-queue-thread-pool" => "pool1")
 ],
 "name" => "count",
 "value" => 20,
 "operation-headers" => {
    "rollback-on-runtime-failure => false
 }
}
```

Поддерживаются следующие заголовки операций:

– «rollback-on-runtime-failure» - логическое значение, необязательно, по умолчанию равно «true». Следует ли отменить операцию, успешно обновляющую модель постоянной конфигурации, если она не может быть применена к среде выполнения. Операции, влияющие на постоянную конфигурацию, применяются в два этапа – сначала к модели конфигурации, а затем к фактически запущенным службам. Если возникает ошибка, связанная с моделью конфигурации, операция будет прервана без изменения конфигурации и не будет предпринята попытка изменения запущенных служб. Однако разрешены операции по изменению модель конфигурации даже в случае сбоя в применении изменения к запущенным службам – в том и только в том случае, если для этого заголовка отката при сбое во время выполнения задано значение «false». Таким образом, этот заголовок описывает только то, что происходит, если возникает проблема с применением операции к запущенному состоянию сервера (например, фактическое увеличение размера пула потоков времени выполнения);

– «rollout-plan» - применимо только к запросам, отправленным на контроллер домена или хост-контроллер. Подробности см. в разделе «Операции с планом развертывания»;

– «allow-resource-service-restart» - логическое значение, необязательное, по умолчанию равно «false». Следует ли выполнять операцию, требующую перезапуска некоторых служб среды выполнения для вступления в силу. Более подробную информацию смотрите в разделе «Обсуждение служб ресурсов» в разделе «Применение обновлений к службам среды выполнения» в разделе «Описание модели управления»;

– «roles» - строка или список строк. Имя(-ена) роли(-ей) RBAC, разрешения для которой(-ых) следует использовать при принятии решений об управлении доступом, а не для ролей, обычно связанных с пользователем, вызывающим операцию. Соблюдается только в том случае, если пользователь обычно связан с ролью со всеми разрешениями (т.е. суперпользователь), что означает, что это может использоваться только для уменьшения разрешений для вызывающего абонента, а не для увеличения разрешений;

– «blocking-timeout» - значение «int» (необязательно) по умолчанию равно 300. Максимальное время в секундах, в течение которого операция должна блокироваться в различных точках в ожидании завершения. Если этот период будет превышен, операция будет откатана. Не представляет собой общее максимальное время выполнения операции; скорее, оно предназначено для того, чтобы служить своего рода мерой безотказной работы, предотвращающей выполнение проблемных операций, которые на неопределенный срок ограничивают ресурсы.

Сложные операции.

Корневой ресурс для домена, хост-контроллера или отдельного сервера предоставит доступ к операции с именем «composite». Эта операция выполняет список других операций как атомарную единицу (хотя требования к атомарности могут быть смягчены). Структура запроса на «составную» операцию имеет ту же фундаментальную структуру, что и простая операция (т.е. имя операции, адрес, параметры в качестве пар ключ-значение).

```
{
 "operation" => "composite",
 "address" => [],
 "steps" => [
  {
    "operation" => "write-attribute",
    "address" => [
      ("profile" => "production"),
      ("subsystem" => "threads"),
      ("bounded-queue-thread-pool" => "pool1")
    ],
    "count" => "count",
    "value" => 20
```

```
  },
  {
    "operation" => "write-attribute",
    "address" => [
      ("profile" => "production"),
      ("subsystem" => "threads"),
      ("bounded-queue-thread-pool" => "pool2")
    ],
    "name" => "count",
    "value" => 10
  }
 ],
 "operation-headers" => {
   "rollback-on-runtime-failure => false
 }
}
```

Операция «composite» принимает один параметр:

– «steps» - список, в котором каждый элемент имеет ту же структуру, что и простой запрос на выполнение операции. В приведенном выше примере каждый из двух шагов изменяет конфигурацию пула потоков для другого пула. Между шагами не обязательно должна быть какая-либо конкретная взаимосвязь. Обратите внимание, что заголовки операций отката при сбое во время выполнения и плана развертывания не поддерживаются для отдельных шагов в составной операции.

```
+
The `rollback-on-runtime-failure` operation header discussed above has
a
particular meaning when applied to a composite operation, controlling
whether steps that successfully execute should be reverted if other
steps fail at runtime. Note that if any steps modify the persistent
configuration, and any of those steps fail, all steps will be
reverted.
Partial/incomplete changes to the persistent configuration are not
allowed.
```

*Операции с планом внедрения.*

Операции, нацеленные на ресурсы уровня домена или хоста, потенциально могут повлиять на несколько серверов. Такие операции могут включать в себя "план развертывания", детализирующий последовательность, в которой операция должна быть применена к серверам, а также политики, определяющие, следует ли отменить операцию, если она не может быть успешно выполнена на некоторых серверах.

Если операция включает в себя план развертывания, то его структура выглядит следующим образом:

```
{
  "operation" => "write-attribute",
  "address" => [
    ("profile" => "production"),
    ("subsystem" => "threads"),
    ("bounded-queue-thread-pool" => "pool1")
  ],
  "name" => "count",
  "value" => 20,
  "operation-headers" => {
    "rollout-plan" => {
      "in-series" => [
      {
        "concurrent-groups" => {
          "groupA" => {
            "rolling-to-servers" => true,
            "max-failure-percentage" => 20
          },
          "groupB" => undefined
        }
      },
      {
        "server-group" => {
          "groupC" => {
            "rolling-to-servers" => false,
            "max-failed-servers" => 1
          }
        }
      },
      {
        "concurrent-groups" => {
          "groupD" => {
            "rolling-to-servers" => true,
            "max-failure-percentage" => 20
          },
          "groupE" => undefined
        }
      }
      ],
      "rollback-across-groups" => true
    }
  }
}
```

Как вы можете видеть, план развертывания — это еще одна структура в разделе заголовки операций. Корневой узел структуры допускает два дочерних узла:

– «in-series» - (список) - список действий, которые необходимо выполнить последовательно, при этом каждое действие должно быть завершено до выполнения следующего шага. Каждое действие включает в себя применение операции к серверам в одной или нескольких группах серверов. Смотрите ниже подробную информацию о каждом элементе списка;

– «rollback-across-groups» - (логический) - указывает, должна ли необходимость отката операции на всех серверах в одной группе серверов вызывать откат во всех группах серверов. Это необязательный параметр, значение по умолчанию равно «false».

Каждый элемент в списке под последовательным узлом должен иметь ту или иную из следующих структур:

– «concurrent-groups» - сопоставление имен групп серверов с политиками, управляющими тем, как операция должна применяться к этой группе серверов. Для каждой группы серверов на карте операция может применяться одновременно. Смотрите ниже подробную информацию о конфигурации политики для каждой группы серверов;

– «server-group» - единственное сопоставление ключа/значения имени группы серверов с политикой, управляющей тем, как операция должна применяться к этой группе серверов. Подробнее о конфигурации политики смотрите ниже (Примечание: нет разницы в выполнении плана между этим и «concurrentgroups» карта с единственной записью).

Политика, управляющая тем, как операция применяется к серверам внутри группы серверов, содержит следующие элементы, каждый из которых является необязательным:

– «rolling-to-servers» - (boolean) - если значение равно «true», операция будет применена к каждому серверу в группе последовательно. Если значение равно «false» или не указано иное, операция будет применена ко всем серверам в группе одновременно;

– «max-failed-servers» - (int) - максимальное количество серверов в группе, на которых может произойти сбой в выполнении операции, прежде чем она будет отменена на всех серверах группы. Значение по умолчанию, если оно не указано, равно нулю; т.е. сбой на любом сервере запускает откат по всей группе;

– «max-failure-percentage» - (значение int от 0 до 100) - максимальный процент от общего числа серверов в группе, на которых может произойти сбой при выполнении операции, прежде чем она будет отменена на всех серверах в группе. Значение по умолчанию, если оно не указано, равно нулю; т.е. сбой на любом сервере запускает откат во всей группе.

Если заданы значения как «max-failed-servers», так и «max-failure-percentage», приоритет имеет «max-failure-percentage».

Если посмотреть на приведенный выше (надуманный) пример, то применение операции к серверам в домене будет выполняться в 3 этапа. Если политика для какой-либо группы серверов инициирует откат операции по всей группе серверов, все остальные группы серверов также будут откатаны. Эти 3 фазы состоят из:

1) для групп серверов «groupA» и «groupB» операция будет применена одновременно. Операция будет применена к серверам в «groupA» последовательно, в то время как все серверы в «groupB» будут обрабатывать операцию одновременно. Если более чем на 20% серверов в «groupA» не удастся применить операцию, она будет отменена для всей группы. Если на каком-либо сервере в «groupB» не удастся применить операцию, она будет отменена для всей группы;

2) как только все серверы в «groupA» и «groupB» будут завершены, операция будет применена к серверам в «groupC». Эти серверы будут обрабатывать операцию одновременно. Если более чем на одном сервере в «groupC» не удастся применить операцию, она будет откатана по всей группе;

3) как только все серверы в группах будут завершены, операция будет применена к группам серверов одновременно. Операция будет применена к серверам в группах последовательно, в то время как все серверы в группах будут выполнять операцию одновременно. Если более чем 20% серверов в группах не смогут применить операцию, она будет отменена для всей группы. Если какой-либо сервер в группах не сможет применить операцию, она будет отменена для всей группы.

*План развертывания по умолчанию*.

Все операции, затрагивающие несколько серверов, будут выполняться с использованием плана развертывания. Однако, указывать план развертывания в запросе на операцию не требуется. Если заголовок операции «План развертывания» не указан, будет создан план по умолчанию. План будет иметь следующие характеристики:

– будет только одна фаза высокого уровня. Операция будет применена одновременно ко всем группам серверов, затронутым операцией;

– в пределах каждой группы серверов операция будет применена ко всем серверам одновременно;

– сбой на любом сервере в группе серверов приведет к откату по всей группе;

– сбой в работе любой группы серверов приведет к откату всех остальных групп серверов.

*Создание и повторное использование плана внедрения*.

Поскольку план внедрения может быть довольно сложным, необходимость каждый раз передавать его в качестве заголовка может стать быстро причиняющий боль. Поэтому вместо этого мы можем сохранить его в модели и затем ссылаться на него, когда захотим его использовать.

Чтобы создать план развертывания, вы можете использовать операцию «Добавить план развертывания» следующим образом:

```
rollout-plan add --name=simple --content={"rollout-plan" => {"in-
series" => [{"servergroup"=> {"main-server-group" => {"rolling-to-
servers" => false,"max-failed-servers"=> 1}}}, {"server-group" =>
{"other-server-group" => {"rolling-to-servers" =>true,"max-failure-
percentage" => 20}}}],"rollback-across-groups" => true}}
```

Это позволит создать план развертывания под названием «simple» в хранилище контента.

```
[domain@192.168.1.20:9999 /] /management-client-content=rollout-
plans/rolloutplan=simple:read-resource
{
   "outcome" => "success",
   "result" => {
     "content" => {"rollout-plan" => {
       "in-series" => [
         {"server-group" => {"main-server-group" => {
           "rolling-to-servers" => false,
           "max-failed-servers" => 1
         }}},
         {"server-group" => {"other-server-group" => {
           "rolling-to-servers" => true,
           "max-failure-percentage" => 20
         }}}
       ],
       "rollback-across-groups" => true
     }},
     "hash" => bytes {
       0x13, 0x12, 0x76, 0x65, 0x8a, 0x28, 0xb8, 0xbc,
       0x34, 0x3c, 0xe9, 0xe6, 0x9f, 0x24, 0x05, 0xd2,
       0x30, 0xff, 0xa4, 0x34
     }
   }
}
```

Теперь вы можете ссылаться на план «roolout» в своей команде, добавив заголовок, подобный этому:

```
deploy /quickstart/ejb-in-war/target/WildBoss Pro-ejb-in-war.war --
all-server-groups--headers={rollout name=simple}
```

### 10.5.4 Формат детализированного ответа на операцию

Как отмечалось ранее, базовый метод, который может использовать пользователь WildBoss Pro 26.1 программного управления API, очень прост:

```
ModelNode execute(ModelNode operation) throws IOException;
```

где возвращаемое значение — это детализированное представление ответа, а операция — это детализированное представление вызываемой операции.

Цель этого раздела - задокументировать структуру возвращаемого значения.

Формат запроса приведен в разделе «Формат запроса на выполнение операции с определенным типом».

*Простые ответы*.

Простые ответы на эти вопросы можно получить с помощью следующих типов операций:

– несоставные операции, предназначенные для одного сервера. (Подробнее о составных операциях читайте ниже);

– несоставные операции, которые нацелены на контроллер домена или подчиненный хост-контроллер и не требуют, чтобы ответчик применял операцию на нескольких серверах и объединял их результаты (например, простое считывание свойства конфигурации домена).

Ответ всегда будет содержать простое логическое поле результата с одним из трех возможных значений:

– «success» - операция выполнена успешно;

– «failed» - операция завершилась неудачей;

– «cancelled» - выполнение операции было отменено. (Это было бы необычным результатом для простой операции, которая, как правило, очень быстро достигает точки выполнения, когда ее уже невозможно отменить).

Остальные поля в ответе будут зависеть от того, была ли операция выполнена успешно.

Ответ на неудачную операцию:

```
{
  "outcome" => "failed",
  "failure-description" => "[JBAS-12345] Some failure message"
}
```

В ответ на успешную операцию будет указано дополнительное поле:

– «result» - возвращаемое значение или не определено для операций «void» или тех, которые возвращают значение «null».

Результат, не являющийся недействительным:

```
{
  "outcome" => "success",
  "result" => {
    "name" => "Brian",
    "age" => 22
  }
}
```

Недействительный результат:

```
{
 "outcome" => "success",
 "result" => undefined
}
```

В ответе на отмененную операцию нет других полей:

```
{
 "outcome" => "cancelled"
}
```

*Заголовки ответов.*

Помимо стандартных полей «outcome», результата и описания сбоя, описанных выше, ответ может также содержать различные заголовки, которые предоставляют дополнительную информацию о результатах операции или об общем состоянии сервера. Заголовки будут дочерним элементом поля с именем «response-headers». Например:

```
{
   "outcome" => "success",
   "result" => undefined,
   "response-headers" => {
      "operation-requires-reload" => true,
      "process-state" => "reload-required"
   }
}
```

Заголовок ответа обычно связан с тем, может ли операция быть применена к целевой среде выполнения без необходимости перезапуска некоторых или всех служб или даже самого целевого процесса. Пожалуйста, ознакомьтесь с разделом «Применение обновлений к службам времени выполнения» раздела «Описание модели управления» для обсуждения основных концепций, связанных с тем, что происходит, если для выполнения операции требуется перезапуск службы.

Текущими возможными заголовками ответа являются:

– «operation-requires-reload» - (boolean) - указывает, что для выполнения конкретной операции, которая привела к получению этого ответа, требуется перезапуск всех служб в процессе, чтобы он вступил в силу во время выполнения. Обычно это значение имеет значение «true»; отсутствие заголовка соответствует значению «false».;

– «operation-requires-restart» - (boolean) - указывает, что для выполнения конкретной операции, которая сгенерировала этот ответ, требуется полный перезапуск процесса, чтобы он вступил в силу во время выполнения. Обычно это значение имеет значение «true»; отсутствие заголовка соответствует значению «false»;

– «process-state» - (enumeration) - предоставляет информацию об общем состоянии целевого процесса. Одно из следующих значений:

– «starting» - этот процесс начинается;

– «running» - процесс находится в обычном запущенном состоянии. Заголовок «process-state» обычно не отображается при этом значении; отсутствие заголовка совпадает со значением «running»;

– «reload-required» - была выполнена некоторая операция (не обязательно эта), которая требует перезапуска всех служб, чтобы изменение конфигурации вступило в силу во время выполнения;

– «restart-required» - была выполнена некоторая операция (не обязательно эта), которая требует полного перезапуска процесса, чтобы изменение конфигурации вступило в силу во время выполнения;

– «stopping» - этот процесс прекращается.

*Основные характеристики сложных операций*.

Составная операция — это операция, которая включает в список более одной простой операции и выполняет их автоматически. Дополнительную информацию смотрите в разделе «Составные операции».

Основные составные ответы обеспечиваются следующими типами операций:

– сложные операции, предназначенные для одного сервера;

– сложные операции, которые предназначены для контроллера домена или подчиненного хост-контроллера и не требуют, чтобы ответчик применял операцию на нескольких серверах и объединял их результаты (например, список простых операций чтения свойств конфигурации домена).

Высокоуровневый формат ответа на базовую сложную операцию в значительной степени совпадает с форматом ответа на простую операцию, хотя существует важное семантическое различие. Для составной операции значение флага результата определяется значением поля заголовка запроса операции «rollback-onruntime- failure» для отката во время выполнения. Если в этом поле указано значение «false» (по умолчанию – «true»), флагом результата будет значение «success», если все шаги были успешно применены к постоянной конфигурации, даже если ни один из шагов составной операции не был успешно применен к среде выполнения.

Отличительной чертой ответа на сложную операцию является поле результата. Во-первых, даже если операция не была успешной, поле результата обычно будет присутствовать (Он не будет отображаться, если произошел какой-то немедленный сбой, который не позволил ответчику даже попытаться выполнить отдельные операции). Во-вторых, содержимое поля результатов будет отображаться в виде карты. Каждая запись на карте будет содержать результат выполнения элемента в параметре «steps» запроса на составную операцию. Ключом для каждого элемента на карте будет строка «step-X», где «X» — это индекс, основанный на единице, для каждого элемента на карте. позиция шага в списке шагов запроса. Таким образом, результат каждой отдельной операции в составной операции будет записан.

Результаты отдельных операций будут иметь тот же базовый формат, что и результаты простых операций, описанные выше. Однако, есть некоторые отличия от случая простой операции, когда флаг результата отдельной операции не установлен. Это связано с тем фактом, что в составной операции отдельные операции могут быть отменены или даже не предприниматься.

Если отдельная операция даже не была предпринята (поскольку общая операция была отменена или, что более вероятно, предыдущая операция завершилась неудачей):

```
{
 "outcome" => "cancelled"
}
```

Отдельная операция, которая завершилась неудачей и была отменена:

```
{
 "outcome" => "failed",
 "failure-description" => "[JBAS-12345] Some failure message",
 "rolled-back" => true
}
```

Отдельная операция, которая сама по себе завершилась успешно, но была отменена из-за сбоя другой операции:

```
{
 "outcome" => "failed",
 "result" => {
   "name" => "Brian",
   "age" => 22
 },
 "rolled-back" => true
}
```

Операция, которая завершилась неудачей и была отменена:

```
{
  "outcome" => "failed",
  "failure-description" => "[JBAS-12345] Some failure message",
  "rolled-back" => true
}
```

Вот пример ответа на успешную двухэтапную комбинированную операцию:

```
{
 "outcome" => "success",
 "result" => [
   {
     "outcome" => "success",
     "result" => {
       "name" => "Brian",
       "age" => 22
     }
   },
   {
     "outcome" => "success",
     "result" => undefined
   }
 ]
}
```

И для неудачной трехэтапной составной операции, когда первый шаг был выполнен успешно, а второй - неудачно, что привело к отмене третьего и откату остальных:

```
{
 "outcome" => "failed",
 "failure-description" => "[JBAS-99999] Composite operation failed;
 see individual
 operation results for details",
 "result" => [
  {
    "outcome" => "failed",
    "result" => {
      "name" => "Brian",
      "age" => 22
    },
    "rolled-back" => true
  },
  {
  "outcome" => "failed",
  "failure-description" => "[JBAS-12345] Some failure message",
  "rolled-back" => true
```

```
    },
    {
    "outcome" => "cancelled"
    }
  ]
}
```

*Ответы нескольких серверов*.

Многосерверные ответы предоставляются операциями, которые нацелены на контроллер домена или подчиненный хост-контроллер и требуют, чтобы ответчик применил операцию к нескольким серверам и объединил их результаты (например, почти все обновления конфигурации домена или хоста).

Операции с несколькими серверами выполняются в несколько этапов.

Сначала может потребоваться применить операцию к модели авторитетной конфигурации, поддерживаемой контроллером домена (для настройки «domain.xml») или хост-контроллером (для настройки «host.xml»). Если на этом этапе происходит сбой, операция автоматически откатывается назад с ответом, подобным этому:

```
{
  "outcome" => "failed",
  "failure-description" => {
    "domain-failure-description" => "[JBAS-33333] Failed to apply X to
    the domain model"
  }
}
```

Если операция была адресована модели домена, то на следующем этапе контроллер домена попросит каждый подчиненный хост-контроллер применить ее к своей локальной копии модели домена. Если контроллер не сделает этого, контроллер домена сообщит всем контроллерам хоста о необходимости отменить изменение, и это изменение также будет отменено локально. Ответ клиенту будет выглядеть следующим образом:

```
{
  "outcome" => "failed",
  "failure-description" => {
    "host-failure-descriptions" => {
      "hostA" => "[DOM-3333] Failed to apply to the domain model",
      "hostB" => "[DOM-3333] Failed to apply to the domain model"
    }
  }
}
```

Если предыдущие этапы завершатся успешно, операция будет передана на все затронутые серверы. Если операция прошла успешно на всех серверах, ответ будет выглядеть следующим образом (в этом примере операция имеет недействительный ответ, следовательно, результат для каждого сервера не определен):

```
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {
    "groupA" => {
      "serverA-1" => {
       "host" => "host1",
       "response" => {
```

```
          "outcome" => "success",
          "result" => undefined
        }
      },
      "serverA-2" => {
       "host" => "host2",
       "response" => {
         "outcome" => "success",
       }
     }
    },
    "groupB" => {
     "serverB-1" => {
       "host" => "host1",
       "response" => {
        "outcome" => "success",
        "result" => undefined
       }
     },
     "serverB-2" => {
     "host" => "host2",
       "response" => {
        "outcome" => "success",
        "result" => undefined
       }
     }
    }
  }
}
```

Необязательно, чтобы операция завершилась успешно на всех серверах, чтобы получить результат «результат» ⇒ «успех». Все, что требуется, — это чтобы она завершилась успешно хотя бы на одном сервере, без того, чтобы политики отката в плане развертывания не вызвали откат на этом сервере. Пример ответа в такой ситуации мог бы выглядеть следующим образом:

```
{
   "outcome" => "success",
   "result" => undefined,
   "server-groups" => {
    "groupA" => {
     "serverA-1" => {
       "host" => "host1",
       "response" => {
         "outcome" => "success",
         "result" => undefined
       }
     },
     "serverA-2" => {
       "host" => "host2",
       "response" => {
         "outcome" => "success",
         "result" => undefined
       }
     }
    },
    "groupB" => {
```

```
    "serverB-1" => {
      "host" => "host1",
      "response" => {
        "outcome" => "success",
        "result" => undefined,
        "rolled-back" => true
      }
    },
    "serverB-2" => {
      "host" => "host2",
      "response" => {
        "outcome" => "success",
        "result" => undefined,
        "rolled-back" => true
      }
    },
    "serverB-3" => {
      "host" => "host3",
      "response" => {
        "outcome" => "failed",
        "failure-description" => "[DOM-4556] Something didn't work
        right",
        "rolled-back" => true
      }
    }
   }
}
```

Наконец, если операция завершится неудачей или будет откатана на всех серверах, пример ответа будет выглядеть следующим образом:

```
{
 "outcome" => "failed",
 "server-groups" => {
  "groupA" => {
   "serverA-1" => {
   "host" => "host1",
    "response" => {
      "outcome" => "success",
      "result" => undefined
    }
   },
   "serverA-2" => {
   "host" => "host2",
    "response" => {
      "outcome" => "success",
      "result" => undefined
    }
   }
  },
  "groupB" => {
   "serverB-1" => {
   "host" => "host1",
    "response" => {
     "outcome" => "failed",
     "result" => undefined,
```

```
      "rolled-back" => true
     }
    },
    "serverB-2" => {
    "host" => "host2",
     "response" => {
      "outcome" => "failed",
      "result" => undefined,
      "rolled-back" => true
     }
    },
    "serverB-3" => {
    "host" => "host3",
     "response" => {
       "outcome" => "failed",
       "failure-description" => "[DOM-4556] Something didn't work
       right",
       "rolled-back" => true
     }
    }
   }
  }
}
```

# 11 Рецепты приготовления CLI

## 11.1 Свойства

### 11.1.1 Добавление, чтение и удаление системных свойств с помощью CLI

Для автономного режима:

```
$ ./bin/jboss-cli.sh --connect controller=IP_ADDRESS
[standalone@IP_ADDRESS:9990 /] /system-property=foo:add(value=bar)
[standalone@IP_ADDRESS:9990 /] /system-property=foo:read-resource
{
"outcome" => "success",
"result" => {"value" => "bar"}
}
[standalone@IP_ADDRESS:9990 /] /system-property=foo:remove
{"outcome" => "success"}
```

Для доменного режима используются те же команды, вы можете добавлять/читать/удалять системные свойства для: всех хостов и экземпляров сервера в домене

```
[domain@IP_ADDRESS:9990 /] /system-property=foo:add(value=bar)
[domain@IP_ADDRESS:9990 /] /system-property=foo:read-resource
[domain@IP_ADDRESS:9990 /] /system-property=foo:remove
```

Хост и его серверные экземпляры

```
[domain@IP_ADDRESS:9990 /] /host=master/system-
property=foo:add(value=bar)
[domain@IP_ADDRESS:9990 /] /host=master/system-property=foo:read-
resource
[domain@IP_ADDRESS:9990 /] /host=master/system-property=foo:remove
```

Только один экземпляр сервера

```
[domain@IP_ADDRESS:9990 /] /host=master/server-config=server-
one/systemproperty=
foo:add(value=bar)
[domain@IP_ADDRESS:9990 /] /host=master/server-config=server-
one/systemproperty=
foo:read-resource
[domain@IP_ADDRESS:9990 /] /host=master/server-config=server-
one/systemproperty=
foo:remove
```

### 11.1.2 Обзор всех системных свойств

Обзор всех системных свойств в WildBoss Pro, включая системные свойства операционной системы и свойства, указанные в командной строке с помощью аргументов «- D», «-P» или «--properties».

Автономный

```
[standalone@IP_ADDRESS:9990 /] /core-service=platform-
mbean/type=runtime:readattribute(
name=system-properties)
```

Домен

```
[domain@IP_ADDRESS:9990 /] /host=master/core-service=platform-
mbean/type=runtime:readattribute(
name=system-properties)
[domain@IP_ADDRESS:9990 /] /host=master/server=server-one/core-
service=platformmbean/
type=runtime:read-attribute(name=system-properties)
```

## 11.2 Конфигурация

### 11.2.1 Перечислите подсистемы

```
[standalone@localhost:9990 /] /:read-children-names(child-
type=subsystem)
{
  "outcome" => "success",
  "result" => [
    "batch",
    "datasources",
    "deployment-scanner",
    "ee",
    "ejb3",
    "infinispan",
    "io",
    "jaxrs",
    "jca",
    "jdr",
    "jmx",
    "jpa",
    "jsf",
    "logging",
    "mail",
    "naming",
    "pojo",
    "remoting",
    "resource-adapters",
    "sar",
    "security",
    "threads",
    "transactions",
    "undertow",
    "webservices",
    "weld"
  ]
}
```

### 11.2.2 Перечислите описание доступных атрибутов и дочерних элементов

Описания, возможные типы и значения атрибутов, разрешение и то, разрешены ли выражения ( $\{ ... } ) из базовой модели, отображаются командой «read-resource-description».

```
/subsystem=datasources/data-source=ExampleDS:read-resource-description
{
  "outcome" => "success",
  "result" => {
   "description" => "A JDBC data-source configuration",
   "head-comment-allowed" => true,
   "tail-comment-allowed" => true,
   "attributes" => {
     "connection-url" => {
       "type" => STRING,
       "description" => "The JDBC driver connection URL",
       "expressions-allowed" => true,
       "nillable" => false,
       "min-length" => 1L,
       "max-length" => 2147483647L,
       "access-type" => "read-write",
       "storage" => "configuration",
       "restart-required" => "no-services"
     },
     "driver-class" => {
       "type" => STRING,
       "description" => "The fully qualified name of the JDBC driver
       class",
       "expressions-allowed" => true,
       "nillable" => true,
       "min-length" => 1L,
       "max-length" => 2147483647L,
       "access-type" => "read-write",
       "storage" => "configuration",
       "restart-required" => "no-services"
     },
     "datasource-class" => {
       "type" => STRING,
       "description" => "The fully qualified name of the JDBC
       datasource
       class",
       "expressions-allowed" => true,
       "nillable" => true,
       "min-length" => 1L,
       "max-length" => 2147483647L,
       "access-type" => "read-write",
       "storage" => "configuration",
       "restart-required" => "no-services"
     },
     "jndi-name" => {
       "type" => STRING,
       "description" => "Specifies the JNDI name for the datasource",
       "expressions-allowed" => true,
       "nillable" => false,
       "access-type" => "read-write",
       "storage" => "configuration",
       "restart-required" => "no-services"
     },
     ...
```

### 11.2.3 Просмотр конфигурации в формате XML для модели домена или хост-модели

Предположим, у вас есть хост, который называется master.

```
[domain@localhost:9990 /] /host=master:read-config-as-xml
```

Только для домена или автономно.

```
[domain@localhost:9990 /] :read-config-as-xml
```

### 11.2.4 Сделайте снимок текущего домена

```
[domain@localhost:9990 /] :take-snapshot()
{
 "outcome" => "success",
 "result" => {
   "domain-results" => {"step-1" => {"name" =>
   "JBOSS_HOME/domain/configuration/domain_xml_history/snapshot/201109
   08-
   165222603domain.xml"}},
   "server-operations" => undefined
 }
}
```

### 11.2.5 Сделайте последний снимок «host.xml» для конкретного хоста

Предположим, у вас есть хост, который называется «master».

```
[domain@localhost:9990 /] /host=master:take-snapshot
{
  "outcome" => "success",
  "result" => {
    "domain-results" => {"step-1" => {"name" =>
    "JBOSS_HOME/domain/configuration/host_xml_history/snapshot/2011090
    8-165640215host.xml"}},
    "server-operations" => undefined
  }
}
```

### 11.2.6 Как получить адрес интерфейса

Атрибут для интерфейса называется «resolved-address». Это атрибут среды выполнения, поэтому по умолчанию он не отображается в «:read-resource». Вам необходимо добавить параметр «include-runtime».

```
./jboss-cli.sh --connect
Connected to standalone controller at localhost:9990
[standalone@localhost:9990 /] cd interface=public
[standalone@localhost:9990 interface=public] :read-resource(include-
runtime=true)
```

```
{
   "outcome" => "success",
   "result" => {
      "any" => undefined,
      "any-address" => undefined,
      "any-ipv4-address" => undefined,
      "any-ipv6-address" => undefined,
      "criteria" => [("inet-address" => expression
      "${jboss.bind.address:127.0.0.1}")],
      "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
      "link-local-address" => undefined,
      "loopback" => undefined,
      "loopback-address" => undefined,
      "multicast" => undefined,
      "name" => "public",
      "nic" => undefined,
      "nic-match" => undefined,
      "not" => undefined,
      "point-to-point" => undefined,
      "public-address" => undefined,
      "resolved-address" => "127.0.0.1",
      "site-local-address" => undefined,
      "subnet-match" => undefined,
      "up" => undefined,
      "virtual" => undefined
   }
}
[standalone@localhost:9990 interface=public] :read-
attribute(name=resolved-address)
{
   "outcome" => "success",
   "result" => "127.0.0.1"
}
```

Аналогично и с доменом, просто укажите путь к экземпляру сервера:

```
[domain@localhost:9990 /] /host=master/server=server-
one/interface=public:readattribute(name=resolved-address)
{
   "outcome" => "success",
   "result" => "127.0.0.1"
}
```

## 11.3 Время выполнения

### 11.3.1 Получите все сведения о конфигурации и времени выполнения из CLI

```
./bin/jboss-cli.sh -c command=":read-resource(include-runtime=true,
recursive=true,recursive-depth=10)"
```

## 11.4 Описание

### 11.4.1 Windows и проблема «Нажмите любую клавишу, чтобы продолжить ...»

Сценарии WildBoss Pro для Windows заканчиваются словами «Нажмите любую клавишу, чтобы продолжить ...». Такое поведение полезно, когда скрипт выполняется

двойным щелчком по скрипту, но не тогда, когда вам нужно вызвать несколько команд из пользовательского скрипта (например, «'bin/jboss-admin.bat --connect command=:shutdown'»).

Чтобы избежать сообщения "Нажмите любую клавишу, чтобы продолжить ...", вам не нужно указывать переменную ПАУЗЫ. Вызовите 'set NOPAUSE=true' в командной строке перед запуском любого скрипта WildBoss Pro 26.1 .bat или включите его в свой пользовательский скрипт перед вызовом скриптов из WildBoss Pro.

## 11.5 Статистика

### 11.5.1 Чтение статистики активных источников данных

```
/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-
resource(includeruntime=true)
/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:read-
resource(includeruntime=true)
```

или

```
/subsystem=datasources/data-source=ExampleDS:read-
resource(includeruntime=true,recursive=true)
```

## 11.6 Развертывание

### 11.6.1 Команда развертывания CLI

В дополнение к устаревшим командам «deploy», «undeploy» и «deploymentinfo», которые остаются неизменными, CLI предлагает команду «deployment», которая должным образом разделяет различные варианты использования, возникающие при управлении развертываниями. Эта команда предлагает более простой интерфейс и должна использоваться при управлении развертываниями. Благодаря команде развертывания будут добавлены новые функции, устаревшие команды развиваться не будут. В этом документе содержится краткое описание возможностей этой команды, введите «help deployment», чтобы отобразить список всех доступных действий, и «*help deployment <action>*» для подробного описания действия.

Действия по развертыванию некоторого контента:

– *deployment deploy-file*: чтобы развернуть файл, расположенный в файловой системе;

– *deployment deploy-url*: чтобы развернуть контент, на который ссылается URL-адрес;

– *deployment deploy-cli-achive*: для развертывания некоторого содержимого используется CLI-архив (cli-файл), расположенный в файловой системе.

Действия по включению некоторых развертываний:

– *deployment enable*: чтобы включить данное отключенное развертывание;

– *deployment enable-all*: чтобы включить все отключенные развертывания.

Действия по отключению некоторых развертываний:

– *deployment disable*: чтобы отключить данное включенное развертывание;

– *deployment disable-all*: чтобы отключить все включенные развертывания.

Действия по отмене развертывания некоторых систем:

– *deployment undeploy*: чтобы отменить развертывание данного развертывания и удалить его содержимое из хранилища;

– *deployment undeploy-cli-archive*: чтобы отменить развертывание некоторого содержимого, используйте CLI-архив (cli-файл), расположенный в файловой системе.

Действия для получения информации о некоторых развертываниях:

– *deployment info*: для отображения информации об одном или нескольких развертываниях;

– *deployment list*: чтобы отобразить все существующие развертывания.

### 11.6.2 Поэтапное развертывание с использованием CLI

Может оказаться желательным поэтапное создание и(или) обновление развертывания WildBoss Pro. В этой главе подробно описывается, как этого можно достичь с помощью инструмента WildBoss Pro CLI.

Шаги по созданию пустого развертывания и добавлению индексного html-файла.

1) Создайте пустое развертывание с именем «my app»:

```
[standalone@localhost:9990 /]
/deployment=myapp:add(content=[{empty=true}])
```

2) Добавить «index.html» в мое приложение:

```
[standalone@localhost:9990 /] /deployment=myapp:add-
content(content=[{input-streamindex=<press TAB>
```

Затем воспользуйтесь завершением, чтобы перейти к вашему файлу «index.html».

3) Укажите целевое имя для «index.html» внутри развертывания и выполните операцию:

```
[standalone@localhost:9990 /] /deployment=myapp:add-
content(content=[{input-streamindex=./index.html, target-
path=index.xhtml}]
```

4) После добавления вашего содержимого вы можете просматривать содержимое развертывания с помощью операции просмотра содержимого:

```
[standalone@localhost:9990 /] /deployment=myapp:browse-
content(path=./)
```

5) Вы можете отобразить (или сохранить) содержимое развернутого файла с помощью команды вложения:

```
attachment display --operation=/deployment=myapp:read-
content(path=index.xhtml)
```

6) Вы можете удалить содержимое из развертывания:

```
/deployment=myapp:remove-content(paths=[./index.xhtml])
```

Типсы:

– операция «add-content» позволяет добавить более одного файла (аргумент «content» — это список сложных типов);

– CLI предлагает завершение для аргументов «путь к просмотру содержимого» и «пути к удалению содержимого»;

– вы можете безопасно использовать операции, использующие присоединенные потоки, в пакетных операциях. В случае пакетных операций потоки присоединяются к составной операции.

**Внимание:** в Windows разделитель путей «\» должен быть экранирован, это является ограничением для клинической обработки сложных типов. При заполнении пути к файлу автоматически экранируются предлагаемые пути.

*Примечания для разработчиков обработчиков операций на стороне сервера*.

Чтобы воспользоваться поддержкой CLI для подключенных файловых потоков и завершения работы файловой системы, вам необходимо правильно структурировать аргументы операции. Шаги по созданию операции, которая получает список файловых потоков, подключенных к операции:

1) определите свой операционный аргумент в виде списка INT (тип значения LIST должен быть типа INT);

2) в описании вашего аргумента добавьте 2 следующих логических дескриптора: «filesystem-path» и «attached-streams».

Когда ваша операция вызывается из командной строки, в качестве аргумента будет автоматически предложено завершение работы с файловой системой. Во время выполнения пути к файловой системе будут автоматически преобразованы в индекс подключенных потоков.

### 11.7 Загрузка файлов с помощью CLI

Некоторые ресурсы управления предоставляют доступ к содержимому файлов в виде потоков. Потоки, возвращаемые операцией управления, присоединяются к заголовкам ответа управления. Вложенная команда CLI (подробное описание этой команды приведено в справке CLI) позволяет отображать или сохранять содержимое подключенных потоков.

– Отображение содержимого файла «server.log»:

```
attachment display --operation=/subsystem=logging/log-
file=server.log:read-resource(include-runtime)
```

– Локальное сохранение файла «server.log»:

```
attachment save --operation=/subsystem=logging/log-
file=server.log:read-resource(include-runtime) --file=./server.log
```

– Отображение содержимого развернутого файла:

```
attachment display --operation=/deployment=myapp:read-
content(path=index.xhtml)
```

– По умолчанию существующие файлы будут сохранены. Используйте опцию «--overwrite», чтобы перезаписать существующий файл.

– Вложение можно использовать в пакетном режиме.

### 11.8 Повторение коллекций

Команда «for» позволяет выполнить итерацию содержимого результата операции. В качестве примера, эту команду можно использовать для отображения содержимого файлов манифеста, присутствующих во всех развернутых приложениях. Например:

```
for deployed in :read-children-names(child-type=deployment)
echo $deployed Manifest content
attachment display --operation=/deployment=$deployed:read-
content(path=META
-INF/MANIFEST.MF)
done
```

Когда выполняется этот блок «for», содержимое всех файлов манифеста отображается в консоли CLI.

Типсы:

– область действия определенной переменной ограничена блоком «for»;

– если переменная с таким же именем уже существует, команда «for» выведет сообщение об ошибке;

– если операция не возвращает список, команда «for» выведет сообщение об ошибке;

– блок «for» можно отбросить и не выполнять, добавив опцию «--discard» в поле готово.

### 11.9 Команды безопасности

CLi предлагает команду безопасности для объединения всех действий по управлению, связанных с безопасностью, в единую команду.

– «security enable-ssl-management»: чтобы включить SSL (elytron SSLContext) для интерфейсов управления. Введите в справке «security enable-ssl-management» для получения полного описания команды.

Помимо других способов настройки SSL, эта команда предлагает интерактивный мастер, который поможет вам настроить SSL путем создания самозаверяющего сертификата. Пример использования мастера:

```
security enable-ssl-management --interactive
Please provide required pieces of information to enable SSL:
Key-store file name (default management.keystore):
Password (blank generated):
What is your first and last name? [Unknown]:
What is the name of your organizational unit? [Unknown]:
What is the name of your organization? [Unknown]:
What is the name of your City or Locality? [Unknown]:
What is the name of your State or Province? [Unknown]:
What is the two-letter country code for this unit? [Unknown]:
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
correct y/n
[y]?
Validity (in days, blank default):
Alias (blank generated):
Enable SSL Mutual Authentication y/n (blank n):n
SSL options:
key store file: management.keystore
distinguished name: CN=Unknown, OU=Unknown, O=Unknown, L=Unknown,
ST=Unknown,
C=Unknown
password: KRzne5s1
validity: default
alias: alias-265e6c6d-ff4e-4b8c-8f10-f015d678eb29
Server keystore file management.keystore, certificate signing request
management.csr
and
certificate file management.keystore.pem will be generated in server
configuration
directory.
Do you confirm y/n :y
```

**Примечание**: как только команда будет выполнена, CLI перезагрузит сервер и снова подключится к нему.

Эта команда также может получить сертификаты от центра сертификации «Let's Encrypt», используя параметр «--let's-encrypt». Помимо упомянутого рабочего процесса, пользователю будет предложено указать дополнительную информацию (например, хранилище ключей учетной записи, учетную запись центра сертификации) для получения сертификата от «Let's Encrypt».

– *security disable-ssl-management*: чтобы отключить SSL (elytron SSLContext) для интерфейсов управления. Введите команду «help security disable-ssl-management» для получения полного описания команды;

– *security enable-ssl-http-server*: чтобы включить SSL (elytron SSLContext) для сервера «undertow». Доступен тот же мастер, что и для действия «enable-ssl-management». Введите команду «help security enable-ssl-http server» для получения полного описания команды.

Эта команда также может получить сертификаты от центра сертификации «Let's Encrypt», используя параметр «--let's-encrypt». Помимо упомянутого рабочего процесса, пользователю будет предложено указать дополнительную информацию (например, хранилище ключей учетной записи, учетную запись центра сертификации) для получения сертификата от «Let's Encrypt».

– *security disable-ssl-http-server*: чтобы отключить SSL (elytron SSLContext) для сервера «undertow». Введите команду «help security disable-ssl-http-server» для получения полного описания команды.;

– *security enable-sasl-management*: чтобы включить аутентификацию SASL (elytron SASL factory) для интерфейсов управления. Вызов этой команды без какой-либо опции приведет к тому, что готовая SASLfactory будет связана с http-интерфейсом. Введите «help security enable»-управление sasl для получения полного описания команды.

Эта команда поддерживает подмножество механизмов SASL, таких как: «EXTERNAL», «DIGEST-MD5», «JBOSS LOCAL- USER», «SCRAM-*», … «CLI completer» предлагает набор механизмов, которые можно правильно настроить с помощью этой команды. Каждый механизм может быть связан с областью файлов свойств, областью файловой системы или областью доверенного хранилища в зависимости от его природы.

**Примечание**: как только команда будет выполнена, CLI перезагрузит сервер и снова подключится к нему.

– *security disable-sasl-management*: чтобы отключить SASL для интерфейсов управления. Если предусмотрен механизм, этот механизм будет удален с фабрики, фабрика останется связанной с интерфейсом. Без этого механизма фабрика больше не будет активна в интерфейсе управления. Введите команду «help security disable-sasl-management» для получения полного описания команды;

– *security reorder-sasl-management*: чтобы изменить порядок в списке механизмов SASL, имеющихся на заводе-изготовителе. Порядок расположения механизмов важен, клиенту отправляется первый из списка. Введите «help security reorder-sasl-management» для получения полного описания команды;

– *security enable-http-auth-management*: чтобы включить HTTP-аутентификацию (elytron HTTP factory) для управляющего http-интерфейса. Вызов этой команды без какой-либо опции приведет к тому, что готовая HTTP-фабрика аутентификации будет связана с http-интерфейсом. Введите «help security enable-http-auth-management» для получения полного описания команды.

Эта команда поддерживает подмножество HTTP-механизмов, таких как: «BASIC», «CLIENT_CERT», «DIGEST», ... «CLI completer» предлагает набор механизмов, которые могут быть правильно настроены с помощью этой команды. Каждый механизм может быть связан с областью файлов свойств, областью файловой системы или областью хранилища доверенных данных в зависимости от его природы.

**Примечание**: как только команда будет выполнена, CLI перезагрузит сервер и снова подключится к нему.

– *security disable-http-auth-management*: чтобы отключить HTTP-аутентификацию для интерфейса управления «http». Если предусмотрен механизм, этот механизм будет удален с фабрики, фабрика останется связанной с интерфейсом. Без этого механизма фабрика больше не будет активна в интерфейсе управления. Введите команду «help security disable-http-auth-management» для получения полного описания команды;

– *security enable-http-auth-http-server*: чтобы включить HTTP-аутентификацию (elytron HTTP factory) для данного домена undertow security. Введите «help security enable-http-auth-http-server» для получения полного описания команды.

Эта команда поддерживает подмножество HTTP-механизмов, таких как: «BASIC», «CLIENT_CERT», «DIGEST», ... «CLI completer» предлагает набор механизмов, которые могут быть правильно настроены с помощью этой команды. Каждый механизм может быть связан с областью файлов свойств, областью файловой системы или областью хранилища доверенных данных в зависимости от его природы.

**Примечание**: как только команда будет выполнена, CLI перезагрузит сервер и снова подключится к нему.

– *security disable-http-auth-http-server*: чтобы отключить HTTP-аутентификацию для данного домена безопасности undertow. Если предусмотрен механизм, этот механизм будет удален из фабрики, фабрика останется связанной с доменом безопасности. Без этого механизма фабрика больше не будет активна в домене безопасности. Введите команду «help security disable-http-auth-http-server» для получения полного описания команды.

## 10.10 Разработка стандартных конфигураций с поддержкой микропрофиля

CLI-скрипт «JBOSS_HOME/docs/examples/enable-microprofile.cli» может быть применен к автономной конфигурации по умолчанию, чтобы добавить поддержку микропрофиля.

Влияние на обновленную конфигурацию:

– добавление микропрофильных подсистем;

– удаление подсистемы безопасности;

– удаление области управления;

– Elytron security используется для управления и точек входа в приложения.

По умолчанию скрипт обновляется «standalone.xml» конфигурация. Благодаря «*config=<config name>*» системное свойство, сценарий может быть применен к другой автономной конфигурации.

**Примечание**: этот скрипт должен быть применен в автономном режиме без запущенного сервера.

– для обновления «standalone.xml» конфигурации сервера:

  – *./bin/jboss-cli.sh --file=docs/examples/enable-microprofile.cli*;

– для обновления других конфигураций автономного сервера:

  – *./bin/jboss-cli.sh --file=docs/examples/enable-microprofile.cli -Dconfig=<standalone-full.xml|standalone-ha.xml|standalone-full-ha.xml>*.