Руководство ПО РАЗВЕРТЫВАНИЮ

ПРОГРАММНОГО ПРОДУКТА

SITEX Модуль интерации

СОДЕРЖАНИЕ

[1 Общие сведения 3](#_Toc450314308)

[2 Разворачивание системы 4](#_Toc450314309)

[2.1 Необходимые программы и компоненты 4](#_Toc450314310)

[2.2 Последовательность установки 4](#_Toc450314311)

[2.3 Состав дистрибутива 4](#_Toc450314312)

[2.4 Установка и настройка общесистемного ПО 4](#_Toc450314313)

[2.4.1 Назначение прав для пользователя, под которым будет проводится установка и настройка ПО 4](#_Toc450314314)

[2.4.2 Установка и настройка Java 5](#_Toc450314315)

[2.4.3 Установка и настройка сервера приложений Tomcat 5](#_Toc450314316)

[2.4.4 Установка и настройка сервера баз данных Postgre SQL 6](#_Toc450314317)

[2.5 Установка и настройка решения 7](#_Toc450314318)

[2.5.1 Необходимые библиотеки 7](#_Toc450314319)

[2.5.2 Необходимые модули 8](#_Toc450314320)

[2.5.3 Генерация стабов клиентской части веб-сервиса БПМС 8](#_Toc450314321)

[2.5.4 Конфигурация модуля интеграции 9](#_Toc450314322)

[*2.5.4.1 Основной конфигурационный файл* 9](#_Toc450314323)

[*2.5.4.2 Конфигурация модуля интеграции параметрами контекста* 10](#_Toc450314324)

[*2.5.4.3 Конфигурация на основе системных переменных* 11](#_Toc450314325)

# Общие сведения

SiTex Модуль интеграции – программное обеспечение, предназначенное для обеспечения обмена электронными сообщениями на основе сведений, полученных из файлов конфигурации, протоколирование произведенных операций, разграничение прав доступа систем, участвующих в обмене электронными сообщениями, создания гибкой архитектуры, формирование подписи ЭП-ОВ (электронная подпись органа власти).

SiTex Модуль интеграции применяется, как транслирующий элемент между различными частями СМЭВ и предназначен для ведения журналирования, разграничения прав доступа систем, создания гибкой архитектуры СМЭВ.

На рисунке показан путь запроса от потребителя к поставщику и его ответа через «сервер интеграции».

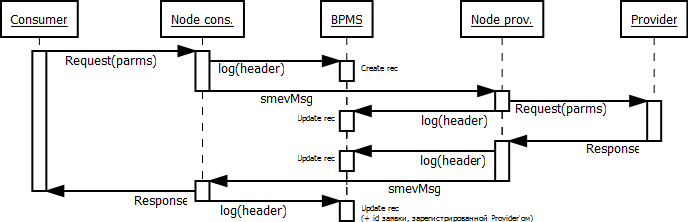


Рисунок 1. Путь запроса от потребителя к поставщику

# Разворачивание системы

## Необходимые программы и компоненты

Для успешной установки и запуска решения необходимо наличие:

* Операционная система семейства Linux;
* Сервер баз данных Postgre SQL;
* Веб сервер ApacheTomcat;
* Java Development Kit.

## Последовательность установки

* Назначение прав для пользователя, под которым будет проводится установка и настройка;
* Установка и настройка Java;
* Установка и настройка сервера приложений Apache Tomcat;
* Установка и настройка сервера баз данных Postgre SQL;
* Установка и настройка решения.

## Состав дистрибутива

* База данных.
* Файлы скриптов для базы данных.
* Дистрибутив Системы.

## Установка и настройка общесистемного ПО

### Назначение прав для пользователя, под которым будет проводится установка и настройка ПО

Для назначения прав для пользователя, под которым будет проводится настройка, необходимо отредактировать файл

В файл /etc/sudoers добавить строку: user ALL=(ALL) ALL

Далее необходимо перезайти в систему под пользователем, который будет использоваться для установки и настройки

### Установка и настройка Java

Необходимо скачать дистрибутив JDK с официального сайта разработчика http://www.oracle.com/technetwork/java/javase/java-archive-downloads-javase6-419409.html#jdk-6u45-oth-JPR

Выберите версию пакета Java-разработчика (Java Development Kit (JDK)) не версии 6 и сохраните файл на диск (для скачивания необходимо иметь учетную запись на сайте [www.oracle.com](http://www.oracle.com)).

Например, для версии Linux x64 можно выбрать файл для скачивания http://download.oracle.com/otn/java/jdk/6u45-b06/jdk-6u45-linux-x64.bin

Запустите мастер установки с следуя его инструкциям проведите установку системы, установку необходимо проводить следуя инструкциям разработчика соответствующего программного обеспечения.

Пример установки:

Ставим java в /var

В файл /home/user/.bash\_profile добавляем строки

export JAVA\_HOME=/var/jdk1.6.0\_45/

export JRE\_HOME=/var/jdk1.6.0\_45/jre

export PATH=$PATH:/var/jdk1.6.0\_45/bin:/var/jdk1.6.0\_45/jre/bin

Проверяем работоспособность командой java -version

### Установка и настройка сервера приложений Tomcat

Необходимо скачать дистрибутив Tomcat версии 7 с официального сайта разработчика <http://tomcat.apache.org/download-70.cgi>

Для примера, скачиваем файл дистрибутива <http://apache-mirror.rbc.ru/pub/apache/tomcat/tomcat-7/v7.0.67/bin/apache-tomcat-7.0.67.zip>

Распаковываем его содержимое в /var

В файл /home/user/.bash\_profile добавляем строку:

export CATALINA\_HOME=/var/apache-tomcat-7.0.67

Даем права на запуск для файлов \*.sh в папке /var/apache-tomcat-7.0.67/bin

Меняем владельца папки /var/apache-tomcat-7.0.67

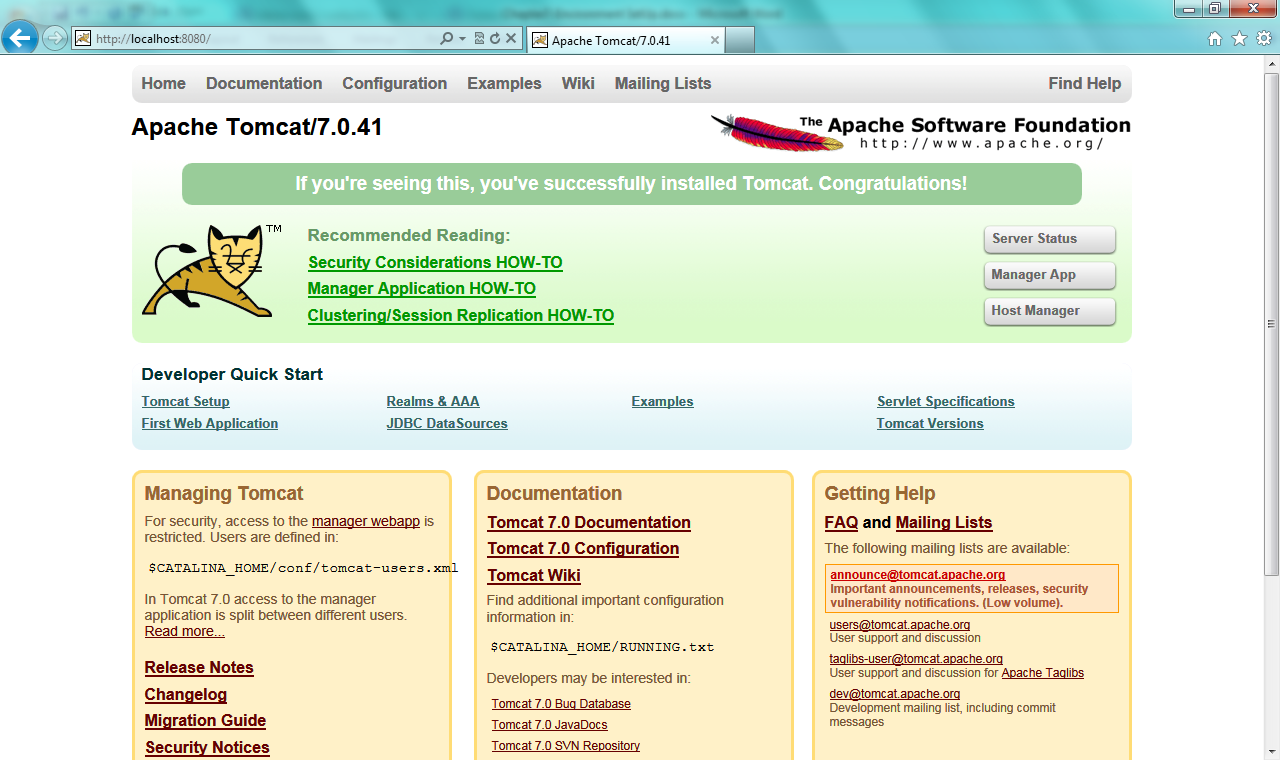
chown -R user:user ./apache-tomcat-7.0.67

При необходимости, если на порту 8080 что-либо запущено, в файле \conf\server.xml меняем параметр сonnector port="8080" например на 8090

Запускаем tomcat с помощью командного файла /var/apache-tomcat-7.0.67/bin startup.sh

Для проверки корректного функционирования Apache Tomcat необходимо используя любой интернет браузер перейти на адрес:    <http://localhost:8080> , где 8080 порт, указанный при установке Tomcat.

В случае корректной работы веб сервера в окне браузера будет доступна домашняя страница Tomcat.



### Установка и настройка сервера баз данных Postgre SQL

Установка и настройка сервера баз данных Postgre SQL будет показана на примере ОС AltLinux 7.

Для установки Postgre SQL в консоли необходимо ввести следуюшие команды:

sudo apt-get install postgresql9.5

sudo apt-get install postgresql9.5-server

sudo apt-get install postgresql9.5-contrib

Для инициализации первоначальных баз данных необходимо выполнить команду:

sudo /etc/init.d/postgresql initdb

Для запуска сервиса необходимо выполнить команду:

sudo service postgresql start

Далее необходимо войти в консоль postgresql и создать пользователя tomcat.

Для входа в консоль необходимо выполнить команду:

sudo -u postgres psql

Для создания пользователя tomcat необходимо выполнить команду (обратите внимание на обязательный символ “;” в конце команды):

CREATE USER tomcat WITH password '12345678';

В случае удачного выполнения вышеуказанной команды в консоли появится сообщение - CREATE ROLE.

Для выхода из консоли необходимо набрать \q и нажать клавишу Enter.

## Установка и настройка решения

### Необходимые библиотеки

Все библиотеки (автоматически загружаются при maven сборке):

activation-1.1.jar  
activemq-all-5.5.1.jar  
antlr-2.7.2.jar  
cloning-1.7.9.jar  
commons-beanutils-1.7.0.jar  
commons-chain-1.1.jar  
commons-collections-20040616.jar  
commons-digester-1.8.jar  
commons-io-1.3.2.jar  
commons-lang-2.6.jar  
commons-logging-1.1.1.jar  
commons-validator-1.3.1.jar  
dom4j-1.1.jar  
jaxb-api-2.2.1.jar  
jaxws-api-1.0.jar  
joda-time-1.6.2.jar  
json-simple-1.1.1.jar  
jstl-1.2.jar  
log4j-1.2.16.jar  
objenesis-1.2.jar  
opensaml-2.5.1-1.jar  
openws-1.4.2-1.jar  
oro-2.0.8.jar  
saaj-api-1.3.4.jar  
slf4j-api-1.5.11.jar  
slf4j-log4j12-1.5.11.jar  
smev-headers-1.0.2.jar  
smev-signer-1.2.jar  
sslext-1.2-0.jar  
standard-1.1.2.jar  
stax-api-1.0-2.jar  
struts-core-1.3.8.jar  
struts-taglib-1.3.8.jar  
struts-tiles-1.3.8.jar  
velocity-1.7.jar  
velocity-tools-2.0.jar  
wsdl4j-1.6.2.jar  
wss4j-1.6.4.jar  
xml-apis-1.0.b2.jar  
xml-security-impl-1.0.jar  
xml-writer-0.2.jar  
xmlsec-1.4.6.jar  
xmltooling-1.3.2-1.jar

### Необходимые модули

smev-headers

smev-signer

### Генерация стабов клиентской части веб-сервиса БПМС

В репозитарии хранится сценарий bpmsStubGenerator.bat, с помощью которого, перед сборкой проекта, необходимо создать необходимые классы клиентской части веб-сервиса БПМС

Вот его содержимое:

|  |
| --- |
| wsimport.exe -Xnocompile -d src -p sx.smev.standalone.modules.ws.stubs.bpms [http://venus:8080/smev\_bpms///sxservices/bpms.jws?wsdl](http://venus:8080/smev_bpms/sxservices/bpms.jws?wsdl) |

Утилита wsimport входит в стандартную поставку jdk 1.6. В параметрах заданы правила создания исходных кодов клиентской части, указан пакет назначения и адрес веб-сервиса БПМС.

### Конфигурация модуля интеграции

#### *Основной конфигурационный файл*

|  |
| --- |
| #BPMS service location  bpms.wsdl=${bpmsWsdl}  #BPMS service namespace  bpms.namespace=${bpmsNamespace}  #Timeout preference for connections in milliseconds  module.connectionTimeout=60000  #Read timeout for connections  module.readTimeout=60000  #Thread count for service loaders  module.service.load.thread.count=1  #Original store directory for wsdl  module.wsdl.original.dir=/wsdl-ext/original  #Processed store directory for wsdl  module.wsdl.processed.dir=/wsdl-ext/processed  #Wsdl and xsd-schema address context  module.mapping.wsdl=/wsdl/  #Endpoint mapping context  module.mapping.service=/service/  #Fsmev endpoint mapping context  module.mapping.service.fsmev=/fsmev/  #Fsmev templates path  module.fsmev.service.templates=/fsmev  #Fsmev templates patameters  module.fsmev.service.templates.param.1.name=extOrderNumber  module.fsmev.service.templates.param.1.expression=//\*[(substring(local-name(), (string-length(local-name()) - string-length('Response')) + 1) = 'Response')]/\*/\*[local-name() = 'guid']  #UidHeader namespace  module.uidh.namespace=http://sys.smev.ru/xsd/uidh  #Fsmev UidHeader comsumer value  module.fsmev.uidh.consumer=rtk  #Web service security header parameters  module.fsmev.wss.user=${moduleFsmevWssUser}  #smev-ppu  module.fsmev.wss.password=${moduleFsmevWssPassword}  #module.fsmev.wss.password=JtBwWrBuk8jdXLyfCGHJ+t0JeCqsAgLBoDnzHMQOWamLIA511Va0tRK+/SbOJoR2CDbrdh6N/+xQkM02tbQV5StyGzW6QC5Se5v0+VfowjRZlVj/XO3UZAaBTOeqqml1RUpDXYqkrlv6dAooNcbm3GAysMDOWbHlZBT3UYa6glQ=  #smev-rkt  #module.fsmev.wss.password=LGRGs8ab+hDU9r7D1/3ZX4JzmAfG0nofxDu0FvtglPlJmgt1owaB4kO5jxulQCSCNMJqS/ i+9Yd1IHZQBOY0QvWtUQxZm+MGYuN4RFsqLbzrdWyTyd6heRT0kZuLjci/aGHfrxLpvsMm AP5bw9RIy71RGWTNuYl9MKRs07Anq08=  #Node code. Using for node journalling in BPMS  module.code=${moduleCode}  # Technical sign store params  module.crypto.alias=${moduleCryptoAlias}  module.crypto.password=${moduleCryptoPassword}  module.crypto.store=HDImageStore  # JMS Parameters for lounraling messages to BPMS  jms.logging.enabled=true  jms.initial.context.factory=org.apache.activemq.jndi.ActiveMQInitialContextFactory  jms.provider.url=${jmsProviderUrl}  jms.topic.nodelog=node.log  jms.connection.factory.names=topicConnectionFactory |

#### *Конфигурация модуля интеграции параметрами контекста*

Может быть прописан в server.xml или в отдельном файле $CATALINA\_HOME\conf\Catalina\localhost\context\_name.xml

|  |
| --- |
| <Context docBase="C:\workspace\sitex\deploySmevNode\" path="/node">  <Parameter name="configFile" value="node.properties" />  </Context> |

#### *Конфигурация на основе системных переменных*

Имеется возможность не хранить все параметры в конфигурационном файле модуля интеграции. Это удобно при поддержке его актуальности. В этом случае в конфигурационном файле модуля интеграции значение параметра записывается в виде: parameter.name=${parameter.name.value}

Далее есть два варианта:

Прописать значение параметра parameter.name.value в System.property. В этом случае при загрузке ноды этот параметр подставится автоматически.

Прописать значение параметра в контексте ноды в server.xml Tomcat. Аналогично тому, как прописан сам конфигурационный файл ноды

|  |
| --- |
| <Context docBase="C:\workspace\sitex\deploySmevNode\" path="/node">  <Parameter name="configFile" value="node.properties" />  <Parameter name="param.name.value" value="value" />  </Context> |