Progress FUSE

FUSE™ ESB

Managing the Container
[DRAFT]

Version 4.1
April 2009

PROGRESS SOFTWARE

Managing the Container

Version 4.1

Publication date 22 Jul 2009

Legal Notices

Table of Contents

List of Tables

List of Examples

Chapter 1. Starting the FUSE ESB Runtime

The default way for deploying the FUSE ESB runtime is to deploy it as a standalone server with an active console. You can also deploy the runtime to run as a background process without a console.

Setting up your environment

You can start the FUSE ESB runtime from the installation directory without doing any work. However, if you want to start it in a different folder you will need to do the following:

1. Add the `bin` directory of your FUSE ESB installation to the `PATH`.

   Windows

   ```
   set PATH=%PATH%;InstallDir\bin
   ```

   ***NIX**

   ```
   export PATH=$PATH,InstallDir/bin
   ```

2. Add a folder called `deploy` to the folder from which you intend to start the FUSE ESB runtime.

   This folder is the hot deployment folder from which bundles and JBI artifacts will be loaded.

   For more information see on page 21.

3. Copy the desired bundles and JBI artifacts from *`InstallDir`*`/deploy` into the new `deploy` folder.

These components will be automatically available when the FUSE ESB runtime starts. You can add more components as they are needed.

Launching the runtime

If you are launching the FUSE ESB runtime from the installation directory use the following command:

Windows

```
bin\servicemix.bat
```

***NIX**

```
bin/servicemix
```

If you are launching it another directory and already performed the steps in "Setting up your environment" on page 9, use the **servicemix** command.

If the FUSE ESB runtime starts up correctly you should see the following on the console:

```
 ____                  _            __  __ _
/ ___|  ___ _ _____   _(_) ___ ___|  \/  (_)_  __
\___ \ / _ \ '__\ \ / / |/ __/ _ \ |\/| | \ \/ /
 ___) |  __/ |   \ V /| | (_|  __/ |  | | |>  <
|____/ \___|_|    \_/ |_|\___\___|_|  |_|_/_/\_\

 ServiceMix (fuse-4.0.0)

Type 'help' for more information.
------------------------------------------------------------
-------------------
servicemix>
```

Warning

By default, remote console access is enabled and largely unsecured. The username and password are not validated. In a production

environment, you should either disable the remote console or configuring real authentication using JAAS.

Launching the runtime without a local console

To launch the FUSE ESB runtime without a local console you use the following command:

Windows

```
bin\servicemix.bat server
```

***NIX**

```
bin/servicemix server
```

If you are launching it another directory and already performed the steps in "Setting up your environment" on page 9, use the **servicemix server** command.

Warning

By default, remote console access is enabled and largely unsecured. The username and password are not validated. In a production environment, you should either disable the remote console or configuring real authentication using JAAS.

Launching the runtime with only a local console

In production environments you may want to have a runtime instance accessible using only a local console. You can do this by launching the runtime in client mode using the following command:

Windows

```
bin\servicemix.bat client
```

***NIX**

```
bin/servicemix client
```

If you are launching it another directory and already performed the steps in "Setting up your environment" on page 9, use the **servicemix client** command.

If the FUSE ESB runtime starts up correctly you should see the following on the console:

```
 ____                  _           __  __ _
/ ___|  ___ _ _____   _(_) ___ ___|  \/  (_)_  __
\___ \ / _ \ '__\ \ / / |/ __/ _ \ |\/| | \ \/ /
 ___) |  __/ |   \ V /| | (_|  __/ |  | | |>  <
|____/ \___|_|    \_/ |_|\___\___|_|  |_|_/_/\_\

 ServiceMix (fuse-4.0.0)

Type 'help' for more information.
------------------------------------------------------------
--------------------
servicemix>
```

Chapter 2. Using the Remote Instances of the Runtime

It does not always make sense to manage an instance of the FUSE ESB runtime using its local console. You can manage the FUSE ESB runtime remotely using the remote console.

Introduction to Using Remote Runtime Instances

Overview

When you start the FUSE ESB runtime in its default mode or in server mode, it enables a remote console that can be accessed from any other FUSE ESB console. The remote console provides all of the functionality of the local console and allows a remote user complete control over the container and the services running inside of it.

Note

When run in client mode the FUSE ESB runtime disables the remote console.

By default, the remote console is configured to use minimal security. When you attempt to connect to a remote FUSE ESB runtime, it will ask you for a username and a password. It does not, however, authenticate the values provided.

You can configure the remote console to use more stringent security. You can also configure the console to use a different port and SSL if required.

Managing remote runtimes

There are two basic ways of managing of remote instances of the FUSE ESB runtime.

- create them as children of a parent runtime and mange the children using the admin shell
- deploy them as separate installations and access each instance using the remote shell

---

Configuring a remote console's address

If you are running multiple instances of the FUSE ESB runtime on the same host or if you want to use SSL to access the remote console, you will need to change the address at which the runtime exposes its remote console. You control the address using the remoteShellLocation property. This property is stored in the *InstallDir*`/etc/org.apache.servicemix.shell.cfg` configuration file.

Example 2.1 on page 15 shows a sample configuration that changes the port used to 8102.

Example 2.1. Changing the Remote Console's Address

```
remoteShellLocation=tcp://0.0.0.0:8102/
```

Securing a remote console

Te easiest way to secure the remote console is to force it to use a secure communication channel over SSL. To do this you can change the remote console's address to use the `ssl://` URL prefix to specify the address as shown in Example 2.2 on page 15.

Example 2.2. Using SSL to Access the Remote Console

```
remoteShellLocation=ssl://0.0.0.0:8101/
```

The remote console uses the *RshServer* realm for authenticating users. The default implementation is a dummy that should be overridden by supplying a JAAS configuration file similar to the one shown in Example 2.3 on page 15.

Example 2.3. Configuring the Remote Console's Security Realm

```
<jaas:config id="RshServer" xmlns:jaas="http://service
mix.apache.org/jaas">
    <jaas:module className="org.foo.SimpleLoginModule"
flags="required">
        property1=joe
        property2=fred
    </jaas:module>
</jaas:config>
```

For more information on configuring security realms see ....

You can also supply your own keystores and truststores by adding the following to the *InstallDir*`/etc/org.apache.servicemix.shell.cfg` configuration file.

Example 2.4. Keystore and Truststore Properties

```
clientKeyAlias=servicemixAlias
clientKeystore=RshKeystore
clientTruststore=RshTruststore
serverKeyAlias=servicemixAlias
serverKeystore=RshKeystore
serverTruststore=RshTruststore
```

Using the Admin Shell

Overview

The admin shell contains commands for creating and managing instances of the FUSE ESB runtime. Each runtime created using the admin shell is a child instance of the runtime that created it. The children are easily managed using names instead of network addresses.

Creating new instances

You create a new runtime instance using the admin shell's **create** command. As shown in Example 2.5 on page 16, the **create** command causes the runtime to create a new runtime installation in the active runtime's `instances/`*`instanceName`*. The newly create instance is a direct copy of its parent. The only difference between parent and child is the port number they listen on. The child instance is assigned a port number based on an incremental count starting at 8101.

Example 2.5. Create a Runtime Instance

```
servicemix>admin create finn
Creating new instance on port 8106 at: /home/fuse/esb4/instances/finn
Creating dir:  /home/fuse/esb4/instances/finn/bin
Creating dir:  /home/fuse/esb4/instances/finn/etc
Creating dir:  /home/fuse/esb4/instances/finn/system
Creating dir:  /home/fuse/esb4/instances/finn/deploy
Creating dir:  /home/fuse/esb4/instances/finn/data
Creating file: /home/fuse/esb4/instances/finn/etc/config.properties
Creating file: /home/fuse/esb4/instances/finn/etc/org.apache.servi
cemix.features.cfg
Creating file: /home/fuse/esb4/instances/finn/etc/org.ops4j.pax.log
ging.cfg
Creating file: /home/fuse/esb4/in
stances/finn/etc/org.ops4j.pax.url.mvn.cfg
Creating file: /home/fuse/esb4/instances/finn/etc/startup.properties
Creating file: /home/fuse/esb4/instances/finn/etc/system.properties
Creating file: /home/fuse/esb4/instances/finn/etc/org.apache.servi
cemix.shell.cfg
Creating file: /home/fuse/esb4/instances/finn/bin/servicemix

servicemix>
```

Change a child's port

If you do not like the port number assigned to a child instance you can change it using the admin shell's **change-port** command. The syntax for the command is:

```
admin change-port instanceName portNumber
```

Important

You can only use the **change-port** command on stopped runtime instances.

Starting child instances

New instances are created in the stopped state. To start a child instance and make it ready to host applications, you use the admin shell's **start** command. The **start** command takes a single argument, *instanceName*, that identifies the child you want started.

Connecting to a child instance

You can connect to a started child instance's remote console using the admin shell's **connect** command.As shown in Example 2.6 on page 17, the **connect** takes three arguments:

Example 2.6. Admin connect Command

```
admin connect { instanceName } {-u username} {-p password}
```

instanceName

This argument specifies the name of the child to which you want to connect.

-u *username*

This argument specifies the username used to connect to the child's remote console. The default value is `smx`.

By default, this value is not authenticated. It is recommended that you configure the console to use better authentication. See "Securing a remote console" on page 15

-p *password*

This argument specifies the password used to connect to the child's remote console. The default value is `smx`.

By default, this value is not authenticated. It is recommended that you configure the console to use better authentication. See "Securing a remote console" on page 15

Stopping a child instance

You can shutdown a running child instance using the admin shell's **stop** command. The **stop** command takes a single argument, *instanceName*, that identifies the child you want stopped.

Destroying a child instance

You can permanently delete a stopped child instance using the admin shell's **destroy** command. The **destroy** command takes a single argument, *instanceName*, that identifies the child you want removed.

Important

You can only remove stopped children.

Managing Remote Instances

Overview

The FUSE ESB command console includes commands that enable you to connect to and start remote instances of the FUSE ESB runtime. The remote instance is not required to be a child of your active runtime console.

Connecting to a remote console

You connect to a remote runtime's console using the **ssh** command. Example 2.7 on page 19 shows the **ssh** command and its arguments:

Example 2.7. ssh Command

```
ssh {-U username} {-P password} {-p port} { hostname }
```

`-U username`

This argument specifies the username used to connect to the remote console. The default value is `smx`.

By default, this value is not authenticated. It is recommended that you configure the console to use better authentication. See "Securing a remote console" on page 15

`-P password`

This argument specifies the password used to connect to the remote console. The default value is `smx`.

By default, this value is not authenticated. It is recommended that you configure the console to use better authentication. See "Securing a remote console" on page 15

`-p port`

This argument specifies the port used to access the desired runtime's remote console. By default this value is `8101`.

You can configure this address by editing the definition of `sshPort` in the configuration file `org.apach.servicemix.shell.cfg`, located in your FUSE ESB installation directory at `install_dir\etc\`. For more information see "Configuring a remote console's address" on page 14

hostname

This argument specifies the hostname of the remote console you want to connect to.

Example

Example 2.8 on page 20 shows an example of the command used to connect to a remote console using the default configuration.

Example 2.8. Connecting to a Remote Console Using the Default URI

```
servicemix>ssh -U smx -P smx -p 8108 remote_host
```

Starting a remote server

You start a remote server using the **sshd** command. Example 2.9 on page 20 shows the **sshd** command and its arguments:

Example 2.9. sshd Command

ssh {-b} {-p *port*}

-b

This option specifies that the server run in the background.

-p *port*

This argument specifies the port on which to listen for SSH connections.

Example

Example 2.10 on page 20 shows an example of the command used to start a remote SSH server to run in the background.

Example 2.10. Starting a Remote SSH Server

```
servicemix>sshd -b -p 8108
```

Chapter 3. Configuring the Hot Deployment System

By default, the FUSE ESB runtime scans a directory for bundles and JBI artifacts to automatically load. You can change the location of this folder and the interval at which the folder is scanned.

Default values

By default, the FUSE ESB runtime looks for a folder named `deploy` in the same folder from which the runtime was launched. For example if you launched the runtime from the root folder of your FUSE ESB installation, the hot deployment folder would be *`InstallDir`*`/deploy`.

The default scan interval is 500 milliseconds.

Specifying the hot deployment folder

You can specify the folder the FUSE ESB runtime monitors by setting the org.apache.servicemix.filemonitor.monitorDir property in the *`InstallDir`*`/etc/config.properties` configuration file. The value is the absolute path of the folder to monitor. If you set the value to **`/home/joe/deploy`**, the runtime will monitor a folder in Joe's home directory.

Controlling the scan interval

By default the FUSE ESB runtime scans a hot deployment folder every 500 milliseconds. To change the interval between scans of the hot deployment folders, you can change the org.apache.servicemix.filemonitor.scanInterval property in the *`InstallDir`*`/etc/config.properties` configuration file. The value is specified in milliseconds.

Example

Example 3.1 on page 21 shows a configuration fragment that sets `/home/smx/jbideploy` as the hot deployment folder and sets the scan interval to 1 second.

Example 3.1. Configuring the Hot Deployment Folders

```
#
# FileMonitor properties
#
org.apache.servicemix.filemonitor.configDir        = ${servicemix.base}/etc
org.apache.servicemix.filemonitor.monitorDir       = ${servicemix.base}/deploy
org.apache.servicemix.filemonitor.generatedJarDir  = ${servicemix.base}/data/generated-
bundles
```

```
org.apache.servicemix.filemonitor.scanInterval     = 1000
```

Chapter 4. Logging Configuration

The FUSE ESB runtime uses log4j as its logging mechanism. You can configure the logging levels for a number of the runtime's systems.

Overview

The FUSE ESB runtime uses log4j as its logging mechanism. Using standard log4j configuration you can customize everything from the logging levels reported by the different runtime components to the format used when publishing the log messages.

The log4j configuration is specified in the `etc/org.ops4j.pax.logging.cfg` configuration file. The default configuration sets the root logger's level to `INFO` and the level of most of the child loggers to `WARN` or `ERROR`. The default configuration defines two appenders. One for the console and one for the log file. The console's appender's threshold is set to `INFO` and the file appender's threshold is set to `DEBUG`.

Important

FUSE ESB only supports the Java properties format of the log4j configuration.

In addition, you can set the logging level for the underlying OSGi framework. This is done by editing a property in the `etc/config.properties` configuration file.

Default loggers

Table 4.1 on page 23 lists the configured loggers and their default logging level.

Table 4.1. Loggers

Logger	Level
root	INFO
org.apache	WARN
org.springframework	WARN
org.jenks	WARN
org.apache.activemq	WARN
org.apache.activemq.transport.discovery	ERROR

Logger	Level
org.apache.servicemix	INFO
org.apache.servicemix.jbi.config	WARN
org.apache.servicemix.jbi.deployment	WARN

Changing the log levels

The default logging configuration sets the logging levels so that the log file will provide enough information to monitor the behavior of the runtime and provide clues about what caused a problem. However, the default configuration will not provide enough information to debug most problems.

The most useful logger to change when trying to debug an issue with FUSE ESB is the root logger. You will want to set its logging level to `DEBUG` as shown in Example 4.1 on page 24.

Example 4.1. Changing Logging Levels

```
# Root logger
log4j.rootLogger=DEBUG, out, osgi:VmLogAppender

...
```

Changing the appenders' thresholds

When debugging a problem in FUSE ESB you may want to change the level of logging information that is displayed on the console.
Example 4.2 on page 24 shows an example of setting the root logger to `DEBUG` but limiting the information displayed on the console to `WARN`.

Example 4.2. Changing the Log Information Displayed on the Console

```
log4j.rootLogger=DEBUG, stdout, osgi:VmLogAppender
...

log4j.appender.stdout.threshold=WARN
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE}
| %-5.5p | %-16.16t | %-32.32c{1} | %-32.32C %4L | %m%n
```

```
...
```

Configuring the OSGi logging level

The logging level of the underlying OSGi framework is controlled by the felix.log.level property in the `etc/config.properties` configuration file. The property can be set to a number between `0` and `4`. The levels match the log levels specified in the OSGi Log Service:

Value	Log Level
`0`	None
`1`(default)	Error
`2`	Warning
`3`	Information
`4`	Debug

More information

For more information on configuring log4j see the log4j manual[1].

[1] http://logging.apache.org/log4j/1.2/manual.html

Chapter 5. Changing the JMX Management Properties

The FUSE ESB container uses JMX for its underlying management features. It is easy to configure the properties used by this service.

Overview

Two of the most commonly changed parts of the FUSE ESB runtime configuration are the RMI port and the JMX URL. Both of these properties are set in the `org.apache.servicemix.management.cfg` system property file.

RMI port configuration

The default value for the RMI registry port is 1099. You can edit the RMI port configuration by changing the value for the rmiRegistryPort property in the `org.apache.servicemix.management.cfg` file.

JMX URL configuration

The default URL used by JMX is `service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi`. You can edit the JMX URL by changing the value for the serviceUrl property in the `org.apache.servicemix.management.cfg` file.

Username and password

You can configure the username and password used to connect to the JMX server. The defaults are both `smx`. To change them you edit the jmxLogin property and the jmxPassword property in in the `org.apache.servicemix.management.cfg` file.

Chapter 6. The FUSE ESB Runtime's Persistent Data

The FUSE ESB container caches information about its state and the artifacts deployed to it. It uses this data to make startup faster. You can configure how this information is stored on your file system.

Data that is Persisted

Overview

By default, the FUSE ESB container stores all of its persistent caches relative to its start location. It will create a `data` folder in the directory from which you launch the container. This folder is populated by folders storing information about the message broker used by the container, the OSGi framework, and the JBI container.

The data folder

The `data` folder is used by the FUSE ESB runtime to store persistent state information. It contains the following folders:

- `arctivemq`: contains persistent data needed by any ActiveMQ brokers that are started by the container.
- `cache`: contains exploded versions of the loaded bundles.
- `generated-bundles`: contains bundles that are generated by the container. Typically these are to support deployed JBI artifacts.
- `jbi`: stores information about the JBI artifacts deployed to the FUSE ESB runtime.
- `log`: contains the log files.
- `txlog`: contains the log files used by the transaction management system.

Persistent JBI information

The `jbi` folder is populated by the FUSE ESB runtime when installing / deploying JBI artifacts in the container. It creates a subdirectory for each artifact deployed. For JBI components the folder's name is generated by the component's name. For JBI service assemblies, the folder's name is the identifier of the bundle generated to support the service assembly.

Configuring How Persistent Data is Stored

Overview

You can configure where the container stores the persistent data it uses. You can also control the performance of the caching system by configuring the buffer size.

The bundle cache location

The location of the bundle cache can be configured by either directly specifying the full path to the cache folder or by configuring a profile. By default, FUSE ESB specifies the full path to the cache folder.

You specify the cache's folder by setting the felix.cache.profiledir property in the `config.properties` configuration file. This property takes the full pathname of the folder into which bundle cache is written. If the folder does not exist, it will be created.

Important

The felix.cache.profiledir takes precedence.

You can also configure where the bundle cache is stored by configuring a cache profile. A cache profile is configured using two properties: felix.cache.dir and felix.cache.profile. Both properties are stored in the `config.properties` configuration file.

The felix.cache.dir property specifies the path name for the top-level bundle cache folder. By default, the top-level bundle cache folder is the `.felix` folder of the user's home directory. For instance, if the user name of the user starting the container is `progress`, the default top-level bundle cache folder would be `/home/progress/.felix` on a Linux system.

The felix.cache.profile property specifies the name of the folder user the top-level bundle cache folder the cache for the particular instance will be stored. For instance if the felix.cache.profile property was set to `widgetServiceContainer`, the bundle cache for this instance of the container would be stored in `/home/progress/.felix/widgetServiceContainer` on a Linux system.

Using a cache profile is a good idea if you want to run multiple instances of the container. The bundle caches can be stored in a centralized location.

The generated-bundle cache location

The generated-bundle cache is where the container caches bundles it creates to support jars that are not supplied as OSGi bundles. You can configure the location of this cache by changing the setting of the org.apache.servicemix.filemonitor.generatedJarDir in the `config.properties` file. This property is the full path of the generated-bundle cache.

Adjusting the bundle cache buffer

The felix.cache.bufsize property controls the size of the buffer used to copy bundles into the bundle cache. Its default value is 4096 bytes.

You can adjust this property by editing its value in the `config.properties` configuration file. The value is specified in bytes.

Chapter 7. Configuring Component Thread Pools

The JBI components included in FUSE ESB use a thread pool to process message exchanges. You can configure each component's thread pool independently.

Overview

The JBI components are multi-threaded. Each one maintains a thread pool that it uses to process message exchanges. These thread pools are configured using three properties that control the minimum number of threads in the pool, the maximum number of threads in the pool, and the depth of the component's job queue.

These properties are specified in the component configuration files in the *InstallDir*`/etc` folder. The component configuration files are named using the scheme *ComponentName*`.cfg`. For example, the configuration file for the JMS file component would be `servicemix-file.cfg`.

The thread pool properties can also be configured using a JMX console.

Important

The component needs to be restarted for changes to take effect.

Thread pool properties

Table 7.1 on page 33 lists the properties used to configure component thread properties.

Table 7.1. Component Thread Pool Properties

Property	Default	Description
threadPoolCorePoolSize	8	Specifies the minimum number of threads in a thread pool. If the number of available threads drops below this limit, the runtime will always create a new thread to handle the job.
threadPoolMaximumPoolSize	32	Specifies the maximum number of threads in a thread pool. Setting this property to -1 specifies that it is unbounded.

Property	Default	Description
threadPoolQueueSize	256	Specifies the number of jobs allowed in a component's job queue.

Thread selection

When a component receives a new message exchange it choose the thread to process the exchange as follows:

1. If the component's thread pool is smaller than the corePoolSize, a new thread is created to process the task.
2. If less than queueSize jobs are in the component's job queue, the task is placed on the queue to wait for a free thread.
3. If the component's job queue is full and the thread pool has less than maximumPoolSize threads instantiated, a new thread is created to process the task.
4. The job is processed by the current thread.

Example

Example 7.1 on page 34 shows the configuration for a component whose thread pool can have between 10 and 200 threads.

Example 7.1. Component Thread Pool Configuration

```
...
threadPoolCorePoolSize = 10
threadPoolMaximumPoolSize = 200
...
```

Appendix A. Configuration Files

Container Configuration

You can configure the FUSE ESB runtime using the following files:

- `config.properties` - the main configuration file for the container
- `system.properties` - specifies java system properties

  Any properties set in this file are available at runtime using `System.getProperties()`.
- `startup.properties` - configures what bundles are started in the container and their start-levels

  Entries take the format *`bundle=start-level`*.
- `org.apache.servicemix.features.cfg` - configures a list of feature repositories to register and a list of features to be automatically installed
- `org.apache.servicemix.management.cfg` - configures the JMX system

  For more information see "Changing the JMX Management Properties" on page 27.
- `org.apache.servicemix.shell.cfg` - configures the properties of remote consoles

  For more information see "Managing Remote Instances" on page 19.
- `org.apache.servicemix.transaction.cfg` - configures the transaction feature
- `org.ops4j.pax.logging.cfg` - configures the logging system

  For more information see "Logging Configuration" on page 23.

- `org.ops4j.pax.url.mvn.cfg` - configures additional URL resolvers

Component Configuration

In addition to the container's configuration files, the *`InstallDir`*`/etc` folder may contain a number of component configuration files. The component configuration files are named using the scheme *`componentName`*`.cfg`. For example, the JMS component's configuration file would be `servicemix-jms.cfg`.

The contents of a component's configuration file is largely component specific. However, each component configuration file contains properties for configuring the thread pool used by the component to process message exchanges.

Index