Persistent Deployable Unit Management

Table of Contents

JAIN SLEE provides a file deployer that greatly simplifies the management of JAIN SLEE deployable unit jars. The deployer:

- Handles the installation of enclosed JAIN SLEE components.
- Automatically activates and deactivates services contained.
- Handles the creation, removal, activation, deactivation, link binding and link unbinding of all Resource Adapter Entities.

All operations are persistent, which means that unlike management done through JMX, these survive server shutdowns.

Persistent Deployable Unit Install

To install a deployable unit jar simply copy the file to *$JBOSS_HOME/server/profile_name/deploy/*, where `profile_name` is the server profile name. Child directories can be used.

Persistent Deployable Unit Uninstall

To uninstall a deployable unit jar simply delete the file.

Beyond Deployable Unit (Un)Install

The file deployer provides additional behavior then simply (un)install deployable unit jars, as done through the JAIN SLEE 1.1 DeploymentMBean:

Service (De)Activation

All services contained in the deployable unit jar are activated after the install, and deactivated prior to uninstall. On service activation, if there is an active service in the SLEE, with same service name and vendor, then it is considered an older version, and the SLEE deactivates it. The deactivation of the old, and activation of the new, is done smoothly in a single operation, allowing service upgrades with no down time.

Dependencies Management

The deployer puts the installation process on hold until all of the component's dependencies are installed and activated. When uninstalling, it waits for all of the components which depend on components inside the deployable unit to be uninstalled. After an install or uninstall, the deployer evaluates all operations on hold.

Deploy-Config Extension

A deployable unit jar may include a deploy-config.xml file in its *META-INF/* directory. This file provides additional management actions for persistent install/uninstall operations:

Resource Adaptor Entity Management

It is possible to specify RA entities, and the container will create and activate the RA entities after the deployable unit is installed. During the uninstall process, the container will deactivate and remove those RA entities.

Resource Adaptor Links Management

It is possible to specify RA links, and the container will bind those after the deployable unit is installed. When uninstalled, the container will unbind those RA links as well. The links are set for resource adapter entities created in the *deploy-config.xml* file.

This file should comply with the following schema:

```
<?xml version="1.0" encoding="UTF-8" ?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="deploy-config">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ra-entity" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="property">
    <xs:complexType>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="type" type="xs:string" use="required" />
      <xs:attribute name="value" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="properties">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="property" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="file" type="xs:string" use="optional" />
    </xs:complexType>
  </xs:element>

  <xs:element name="ra-entity">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="properties" maxOccurs="1" minOccurs="0"/>
        <xs:element ref="ra-link" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="resource-adaptor-id" type="xs:string" use="required" />
      <xs:attribute name="entity-name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="ra-link">
    <xs:complexType>
      <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

</xs:schema>
```

```
<ra-entity
    resource-adaptor-
id="ResourceAdaptorID[name=JainSipResourceAdaptor,vendor=net.java.slee.sip,version=1.2
]"
    entity-name="SipRA">
    <properties>
        <property name="javax.sip.PORT" type="java.lang.Integer" value="5060" />
    </properties>
    <ra-link name="SipRA" />
</ra-entity>
```

The deploy-config.xml example above defines a resource adaptor entity named `SipRa`, to be created for the resource adaptor with id `ResourceAdaptorID[name=JainSipResourceAdaptor, vendor=net.java.slee.sip, version=1.2]`, and with a single config property named `javax.sip.PORT` of type `java.lang.Integer` and with value `5060`. Additionally, a resource adaptor link named `SipRa` should be bound to the resource adaptor entity.

After the deployable unit is installed, the resource adaptor entity is created, activated and the resource adaptor link is bound. Before the deployable unit is uninstalled, or the server is shutdown, the link is unbound, then the resource adaptor entity is deactivated, and finally the same resource adaptor entity is removed.

The SLEE includes a deploy-config.xml file at *$JBOSS_HOME/server/profile_name/deploy/restcomm-slee*, where `profile_name` is the server profile name, and that file can be used to specify RA entities and links too. If an RA is installed using the persistent deployer, then SLEE reads its own deploy-config.xml file, and if there are RA entities and/or links specified with same RA ID, then the operations to create/activate/deactivate/remove these are executed too, as if these were specified in the Deployable Unit's own deploy-config.xml.