Using Celtix Configuration

Table of Contents

Overview

This document describes how to write and use a Celtix configuration file.

A Celtix configuration file is actually a Spring Framework XMLBeanFactory configuration file (see `http://www.springframework.org/docs/reference/beans.html#beans-factory` and `http://www.springframework.org/`). However, other than learning how to write the configuration file, you do not need to know anything about the Spring Framework or its APIs.

Illustration 1 shows the hierarchical organization of configurable components within Celtix. The shaded components may be configured through use of a Celtix configuration file. Note that within the client application, configuration settings may be applied at the level of the service (that is, to all invocations that use a specific proxy instance), port, or transport. Within the server application, configuration settings may be applied at the level of the endpoint (that is, to all invocations against a specific service) or transport.

In a later section of this document, you will see how an understanding of this hierarchy is used to identify the component to which a collection of configuration settings should be applied.

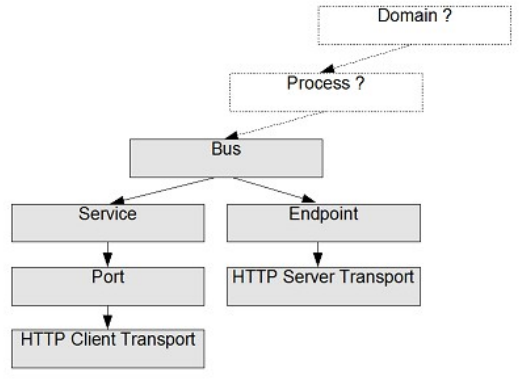

Illustration 1: Configurataion Hierarchy

This document is based on the Celtix Milestone 3 release (December 2005), and there may be revisions to subsequent releases. Refer to the Celtix configuration Wiki page for the most current description of this functionality (**https://wiki.objectweb.org/celtix/Wiki.jsp?page=ConfigurationDocumentation**).

Structure of a Celtix Configuration File

A Celtix configuration file has the syntax shown in the following fragment:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans SYSTEM
      "http://celtix.objectweb.org/configuration/spring/celtix-spring-beans.dtd">

<beans xmlns:...>
  <bean id="..." class="...">
      <property name="...">
          <value>
              ...
          </value>
      </property>
  </bean>
</beans>
```

With the exception of the URL listed in the **DOCTYPE** declaration, this syntax derives directly from the Spring Framework XMLBeanFactory configuration file. Starting from this basic framework, your task is to fill in the entries indicated by the ellipsis (...).

In the opening **beans** element, you will include namespace declarations for the schema files that define acceptable configuration entries. The file may then include one, or more, **bean** elements, each of which corresponds to a configurable component within Celtix.

The value you supply for the **id** attribute generally includes a Bus identifier and the name of the service, or the service and port, to which the configuration applies. There are exceptions to this rule that will be discussed in a later section of this document. The **id** attribute represents the component to which the configuration will be applied, for example, a Bus, a service, a port, or a transport. The **class** attribute provides the class name of a bean within the Spring Framework infrastructure that is responsible for managing the configuration entries.

Finally, the **property** element corresponds to a configuration entry, identified by the **name** attribute, and the content within the **value** elements is the value for that configuration entry.

Filling in values for the **id** attribute and the **class** attribute is fairly straight-forward. The information needed to specify the id is contained in the WSDL file for the service, whereas the information needed to create the class name is derived from the namespace URI of the metadata XML file that defines acceptable configuration entries for the component being configured.

To complete the information in the **property** element and the **value** element you use the types that are defined in multiple schema and metadata XML files, which are described in the following section.

The Celtix Configuration Schema and Metadata XML Files

Celtix provides four schema files that define types typically used in configuration specifications. Additionally, there are seven XML files that describe the metadata needed to identify and set a configuration variable. In both the Celtix binary and source distributions, copies of these files exist within the `celtix.jar` file located in the directory `CELTIX_HOME/lib` and in the `CELTIX_HOME/resources` directory. This document's appendix also includes copies of each of these files; when you are actually writing a Celtix configuration file you should refer to the copies of these files that ship with your actual product as these files may have been revised after this document was written.

The Celtix Schema Files

The file `resources/schemas/configuration/std-types.xsd` includes global element definitions for the most commonly used standard XML datatypes as well as type and global element definitions for some non-standard types, for example, a string list and a list of class:namespace mappings. You may find it convenient to use some of these types, but their use may not be required.

The file `resources/schemas/configuration/jaxws-types.xsd` includes definitions of the types and elements you will use when writing configuration details for a Celtix handler. With Celtix, you can use configuration to specify a handler used in either a client or server application, or to override handler specifications defined within `@HandlerChain` annotations in your source code.

The schema `resources/schemas/configuration/metadata.xsd` describes the syntax of the XML files that Celtix uses to describe the configuration metadata for each configurable component. This is the schema that describes the structure of the XML files described in the following section.

Finally, the schema `resources/schemas/configuration/security.xsd` describes the types and elements you will use when configuring authorization and SSL.

The Celtix XML Metadata Files

The content of each of these XML metadata files adheres to the structure defined in the schema file `resources/schemas/configuration/metadata.xsd`. Each entry within a `configItem` element represents a configurable variable for the component.

The file `resources/config-metadata/bus-config.xml` defines the metadata for configuring the Bus component. This file describes two configurable variables: `bindingFactories` and `transportFactories`. Note that the namespace assigned to this file's content is `http://celtix.objectweb.org/bus/bus-config`. When writing a configuration file that includes Bus related configuration entries, the value of the `class` attribute in the `bean` element is derived from this namespace declaration: `org.objectweb.celtix.bus.bus_config.spring.BusConfigBean`.

The file `resources/config-metadata/endpoint-config.xml` defines the metadata for configuring the endpoint (server) component. This file describes one configurable variable: `handlerChain`. Note that the namespace assigned to this file's content is `http://celtix.objectweb.org/bus/jaxws/endpoint-config`. When writing a configuration file that includes endpoint related configuration entries, the value of the `class` attribute in the `bean` element is: `org.objectweb.celtix.bus.jaxws.endpoint_config.spring.EndpointConfigBean`.

The file `resources/config-metadata/service-config.xml` defines the metadata for configuring the service (client) component. This file describes one configurable variable: `handlerChain`. Note that the namespace assigned to this file's content is `http://celtix.objectweb.org/bus/jaxws/service-config`. When writing a configuration file that includes endpoint related configuration entries, the value of the `class` attribute in the `bean` element is: `org.objectweb.celtix.bus.jaxws.service_config.spring.ServiceConfigBean`.

The file `resources/config-metadata/port-config.xml` defines the metadata for configuring the endpoint's port component. This file describes three configurable variables: `address`, `bindingId`, and `transportId`. Note that the namespace assigned to this file's content is `http://celtix.objectweb.org/bus/jaxws/port-config`. When writing a configuration file that includes endpoint related configuration entries, the value of the `class` attribute in the `bean` element is: `org.objectweb.celtix.bus.jaxws.port_config.spring.PortConfigBean`.

The file **resources/config-metadaata/http-client-config.xml** defines the metadata for configuring the service (client) transport. This file describes many configurable variables. The content of this file is derived from the schema file **resources/schemas/wsdl/http-conf.xsd**, which describes the attributes that may be set on the HTTP transport. Note that the namespace assigned to this file's content is **http://celtix.objectweb.org/bus/transports/http/http-client-config**. When writing a configuration file that includes endpoint related configuration entries, the value of the **class** attribute in the **bean** element is:
org.objectweb.celtix.bus.transports.http.http_client_config.spring.HttpClientConfigBean.

The file **resources/config-metadata/http-server-config.xml** defines the metadata for configuring the endpoint (server) transport. This file describes many configurable variables. The content of this file is derived from the schema file **resources/schemas/wsdl/http-conf.xsd**, which describes the attributes that may be set on the HTTP transport. Note that the namespace assigned to this file's content is **http://celtix.objectweb.org/bus/transports/http/http-server-config**. When writing a configuration file that includes endpoint related configuration entries, the value of the **class** attribute in the **bean** element is:
org.objectweb.celtix.bus.transports.http.http_server_config.spring.HttpServerConfigBean.

The file **resources/config-metadata/http-listener-config.xml** defines the metadata for configuring the endpoint (server) port. The content of this file is derived from the schema files **resources/schemas/wsdl/http-conf.xsd** and **resources/schemas/configuration/security.xsd**. The namesapce assigned to this file's content is **http://celtix.objectweb.org/bus/transports/http/http-listener-config**. When writing a configuration file that includes these entries, the value of the **class** attribute in the **bean** element is:
org.objectweb.celtix.bus.transports.http.http_listener_config.spring.HttpListenerConfigBean.

Writing a Celtix Configuration File

Review the structure of a Celtix configuration file (see page 2). You will write a separate configuration file for each process that you want to configure. You will then define a **bean** element for each component that requires configuration, which means that you must provide a value for the **id** and **name** attributes in the **bean** element, and for the **name** attribute in one, or more, **property** elements. Within the **property** element, you specify a value for this configurable entry. Entering the value is the most difficult part of this process as you must use the information in the schema and XML metadata files as guides to the proper syntax.

The Namespace Declarations

Within the opening **beans** element you should include namespace declarations corresponding to the schema files **org/objectweb/celtix/configuration/config-metadata/types.xsd** and **org/objectweb/celtix/configuration/config-metadata/metadata.xsd**. These declarations are then available to all **bean** elements within the configuration file. The beginning of each Celtix configuration file is shown in the following fragment.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans SYSTEM
      "http://celtix.objectweb.org/configuration/spring/celtix-spring-beans.dtd">

<beans
    xmlns:ct="http://celtix.objectweb.org/configuration/types"
    xmlns:jaxws-types="http://celtix.objectweb.org/bus/jaxws/configuration/types"
>
```

The class Attribute

In the section The Celtix XML Metadata Files, you saw how the namespace used in the metadata XML file is mapped to the name of a Java bean class within the Spring Framework. This class name is then used as the value of the **class** attribute within a **bean** element. The information in the metadata XML and Celtix configuration files is used by the Celtix runtime to instantiate and initialize a bean instance that will manage a component's configuration. The following table summarizes the class attribute values that correspond to each configurable component.

Component	Bean Class Name
Bus	`org.objectweb.celtix.bus.bus_config.spring.BusConfigBean`
Endpoint	`org.objectweb.celtix.bus.jaxws.endpoint_config.spring.EndpointConfigBean`
Service	`org.objectweb.celtix.bus.jaxws.service_config.spring.ServiceConfigBean`
Port	`org.objectweb.celtix.bus.jaxws.port_config.spring.PortConfigBean`
HTTP Client Transport	`org.objectweb.celtix.bus.transports.http.http_client_config.spring.HttpClientConfigBean`
HTTP Server Transport	`org.objectweb.celtix.bus.transports.http.http_server_config.spring.HttpServerConfigBean`
HTTP Listener	`org.objectweb.celtix.bus.transports.http.http_listener_config.spring.HttpListenerConfigBean`

The id Attribute

While the value of the `id` attribute also indicates what component the configuration applies to, it is not a simple one-to-one relationship as is the `class` attribute. This is because the id is generally derived from information in the WSDL file that describes the service. This can be illustrated using the simple hello world WSDL file that follows. The entries you need to specify the id are indicated in bold face font: the targetNamespace, the service name, and the port name. You also need the name assigned to the Celtix Bus. By default, in a simple application where the Bus instance is created transparently by the Celtix runtime, the Bus name is `celtix`.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="HelloWorld"
    targetNamespace="http://objectweb.org/hello_world"
    ...>
    <wsdl:types>
        <schema targetNamespace=
                "http://objectweb.org/hello_world_soap_http/types"
                ...>
            ...
        </schema>
    </wsdl:types>

    <wsdl:message name=...>
        ...
    </wsdl:message>
      .
      .
      .
    <wsdl:message name=...>
        ...
    </wsdl:message>

    <wsdl:portType name=...>
        <wsdl:operation name=...>
            <wsdl:input message=... name=.../>
            <wsdl:output message=... name=.../>
        </wsdl:operation>
          .
          .
          .
        <wsdl:operation name=...>
            <wsdl:input message=... name=.../>
            <wsdl:output message=... name=.../>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name=... type=...>
        <soap:binding style="document"
```

```
                    transport="http://schemas.xmlsoap.org/soap/http"/>

        <wsdl:operation name=...>
            <soap:operation soapAction="" style="document"/>
            <wsdl:input name=...>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name=...>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
          .
          .
          .
        <wsdl:operation name=...>
            <soap:operation soapAction="" style="document"/>
            <wsdl:input name=...>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name=...>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="SOAPService">
        <wsdl:port binding=... name="SoapPort">
            <soap:address location="http://localhost:9000/SoapContext/SoapPort"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

The following table lists which pieces of information are needed to create the id for each configurable component and shows the resulting value specific to this WSDL file.

Component	***Instance Identifier***	***id Value***
Bus	Bus name	`celtix`
Endpoint	Bus name and QName of service	`celtix.{http://objectweb.org/hello_world}SOAPService`
Service	Bus name and QName of service	`celtix.{http://objectweb.org/hello_world}SOAPService`
Port	Bus name, QName of service, and name of port	`celtix.{http://objectweb.org/hello_world}SOAPService.SoapPort`
HTTP Client Transport	Bus name, QName of service, name of port, and string constant	`celtix.{http://objectweb.org/hello_world}SOAPService.SoapPort http-client`

Component	Instance Identifier	id Value
HTTP Server Transport	Bus name, QName of service, name of port, and string constant	`celtix.{http://objectweb.org/hello_world}SOAPService.SoapPort http-server`
HTTP Listener	Bus name, string constant, and port number (where port number is the TCP/IP port number set in the server mainline code).	`celtix.http-listener.port_number`

The property Element

Perhaps the best way to learn how to complete the rest of the Celtix configuration file is to look as some examples.

Setting the URL through configuration

The most straight-forward example is the situation in which you want configure the client application so that the URL used to invoke on the endpoint is defined in the configuration file rather than through the WSDL file's content. For the simple hello world example described above, the following fragment shows the corresponding **`property`** element, which corresponds to the configuration entry that will set the URL.

```
<bean id="celtix.{http://objectweb.org/hello_world}SOAPService.SoapPort"
        class="org.objectweb.celtix.bus.jaxws.port_config.spring.PortConfigBean"
>
      <property name="address">
          <value>
              <ct:stringValue>
                http://localhost:9002/SoapContext/SoapPort
              </ct:stringValue>
          </value>
      </property>

</bean>
```

Since you are configuring an HTTP port, the appropriate XML metadata file to use as a guide is **`resources/config-metadata/port-config.xml`**, and the desired **`configItem`** element is address. Assign the value of the configItem's **`name`** element to the **`property`** element's **`name`** attribute. Within the **`value`** element, you will enter a string that is the desired URL. Notice that the element used to delimit the URL correspond to an element type defined in the schema file **`resources/schemas/configuration/std-types.xsd`**. The configuration scheme also supports a shorthand notation that eliminates the **`stringValue`** elements.

```
<value>http://localhost:9002/SoapContext/SoapPort</value>
```

Specifying a Handler Through Configuration

A more complex example is contained in the server configuration file, **`celtix-server.xml`**, which is included in the handlers product demo.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans SYSTEM
      "http://celtix.objectweb.org/configuration/spring/celtix-spring-beans.dtd">

<beans
    xmlns:ct="http://celtix.objectweb.org/configuration/types"
    xmlns:jaxws-types="http://celtix.objectweb.org/bus/jaxws/configuration/types"
>

  <bean id="celtix.{http://www.objectweb.org/handlers}AddNumbersService"
        class=
      "org.objectweb.celtix.bus.jaxws.endpoint_config.spring.EndpointConfigBean">

      <property name="handlerChain">
          <value>
              <jaxws-types:handlerChainValue>
                 <jaxws-types:handler>
                      <jaxws-types:name>
                          File Logging Handler
                      </jaxws-types:name>
                      <jaxws-types:className>
                          demo.handlers.common.FileLoggingHandler
                      </jaxws-types:className>
                 </jaxws-types:handler>
              </jaxws-types:handlerChainValue>
          </value>
      </property>

  </bean>
</beans>
```

Since you are configuring a service endpoint, the appropriate XML metadata file to use as a guide is **resources/config-metadata/endpoint-config.xml**, and the desired **configItem** element is handlerChain. Assign the value of the configItem's **name** element to the **property** element's **name** attribute. Within the **value** element, you will make entries that identify the class within your application that implements the handler you want to deploy.

Deciding what to place within the **value** element is a little more complex than in the previous example. Look at the schema file **resources/config-metadata/endpoint-config.xml** and note that the handlerChain entry corresponds to the handlerChainType defined in the schema file **resources/schemas/configuration/jaxws-types.xsd**. However, since the handlerChainType is a complex type, and the configuration file must include an element type, you identify the **handlerChainValue** element type as a suitable replacement. Then you look up the composition of the handlerChainType and determine that it is a sequence of zero or more handlerType instances.

```
      <xs:complexType name="handlerChainType">
            <xs:sequence>
                  <xs:element name="handler" type="tns:handlerType"
                              minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
```

And the handlerType is also a sequence in which only the second element is required.

```
      <xs:complexType name="handlerType">
            <xs:sequence>
                  <xs:element name="name" type="xs:string" minOccurs="0"/>
                  <xs:element name="className" type="xs:string"/>
                  <xs:element name="init-param" type="tns:handlerInitParamType"
                              nillable="false" minOccurs="0"
                              maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
```

It is the value assigned to the **className** element that identifies the handler class.

```
                    <jaxws-types:className>
                        demo.handlers.common.FileLoggingHandler
                    </jaxws-types:className>
```

What if your handler class requires initialization parameters. The handlerType allows for this through its **init-param** element, which is an instance of handlerInitParamType. The handlerInitParamType is a

sequence of name value pairs that correspond the handler's initialization values. Since the handlerInitParamType is an unbounded sequence, your configuration entry may have as many **init-param** elements as required. The following fragment illustrates how the **<init-param>** elements would be included in the configuration file.

```
    <property name="handlerChain">
        <value>
            <jaxws-types:handlerChainValue>
               <jaxws-types:handler>
                    <jaxws-types:name>...</jaxws-types:name>
                    <jaxws-types:className>...</jaxws-types:className>

                    <jaxws-types:init-param>
                        <jaxws-types:param-name>
                            arg1
                        </jaxws-types:param-name>
                        <jaxws-types:param-value>
                            value1
                        </jaxws-types:param-value>
                    </jaxws-types:init-param>

                    <jaxws-types:init-param>
                        <jaxws-types:param-name>
                            arg2
                        </jaxws-types:param-name>
                        <jaxws-types:param-value>
                            value2
                        </jaxws-types:param-value>
                    </jaxws-types:init-param>

                </jaxws-types:handler>
            </jaxws-types:handlerChainValue>
        </value>
     </property>
```

Setting Authorization Information

Since you are configuring the client service, the appropriate XML metadata file to use as a guide is **resources/config-metadaata/http-client-config.xml**, and the desired **configItem** element is authorization. Assign the value of the configItem's **name** element to the **property** element's **name** attribute. Within the **value** element, you will make entries that identify the class within your application that implements the handler you want to deploy.

Look at the schema file **resources/config-metadaata/http-client-config.xml** and note that the authorization entry corresponds to the AuthorizationPolicy type defined in the schema file **resources/schemas/configuration/security.xsd**. However, since the AuthorizationPolicy type is a complex type, and the configuration file must include a element type, you identify the **authorization** element as a suitable replacement. Then you look up the composition of the AuthorizationPolicy type and determine that it is a sequence of elements: **UserName**, **Password**, **AuthorizationType**, and **Authorization**.

```
   <xs:complexType name="AuthorizationPolicy">
        <xs:sequence>
            <xs:element name="UserName" type="xs:string" minOccurs="0"/>

            <xs:element name="Password" type="xs:string" minOccurs="0"/>

            <xs:element name="AuthorizationType" type="xs:string" minOccurs="0"/>

            <xs:element name="Authorization" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="authorization" type="tns:AuthorizationPolicy"/>
```

Combining this information, leads to the following **bean** element. Note the use of the **sec:** namespace prefix. You must include the corresponding namespace declaration at the beginning of the configuration file.

```
<beans
```

```
    xmlns:ct="http://celtix.objectweb.org/configuration/types"
    xmlns:sec="http://celtix.objectweb.org/bus/configuration/security">

  <bean id="celtix.{http://objectweb.org/hello_world_soap_http}SOAPService.
                                                          SoapPort.http-client"
        class="org.objectweb.celtix.bus.transports.http.http_client_config.
                                                  spring.HttpClientConfigBean">

      <property name="authorization">
          <value>
              <sec:authorization>
                  <sec:UserName>User</sec:UserName>
                  <sec:Password>celtix</sec:Password>
              </sec:authorization>
          </value>
      </property>

  </bean>
</beans>
```

Setting Transport Attributes

The XML metadata file **resources/config-metadaata/http-client-config.xml**, using the **configItem** httpClient, indicates that the HTTPClientPolicy type can be used to set transport attributes. The HTTPClientPolicy type, defined in the schema file **resources/schemas/wsdl/http-conf.xsd**, is a complex type consisting of multiple attributes. This example illustrates how to use attributes to send the request to a proxy server.

```
    <xs:complexType name="HTTPClientPolicy">
        <xs:annotation>
            <xs:documentation>HTTP client configuration properties.
            Used for configuring a HTTP client port.
            </xs:documentation>
        </xs:annotation>

        <xs:complexContent>
            <xs:extension base="wsdl:tExtensibilityElement">

                <!-- Other attribute definitions -->

                <xs:attribute name="AutoRedirect" type="xs:string" use="optional"
                              default="false"/>

                <!--Proxy server attributes-->
                <xs:attribute name="ProxyServer" type="xs:string" use="optional">
                    <xs:annotation>
                        <xs:documentation>
                            Address of proxy server, if used
                            (proxy servers are a special kind of firewall)
                            proxy.mycompany.com
                        </xs:documentation>
                    </xs:annotation>
                </xs:attribute>

                <xs:attribute name="ProxyServerPort" type="xs:int"
                              use="optional">
                    <xs:annotation>
                        <xs:documentation>
                            Port number of proxy server.
                        </xs:documentation>
                    </xs:annotation>
                </xs:attribute>

                <xs:attribute name="ProxyServerType"
                              type="http-conf:proxyServerType"
                              use="optional" default="HTTP">
                    <xs:annotation>
                        <xs:documentation>
                            Type of number of proxy server.
                        </xs:documentation>
```

```
                    </xs:annotation>
                </xs:attribute>

            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="client" type="http-conf:HTTPClientPolicy"/>
```

The client element may be used to reference the HTTPClientPolicy type in a Celtix configuration file. Note that the bean class used to configure transport attributes is the same as the bean class used to configure authorization. The **`http-conf.xsd`** schema file is described in the namespace **`http://celtix.objectweb.org/transports/http/configuration`**, and you must include a prefix definition for this namespace at the beginning of the Celtix configuration file.

```
<beans
    xmlns:ct="http://celtix.objectweb.org/configuration/types"
    xmlns:sec="http://celtix.objectweb.org/bus/configuration/security"
    xmlns:http-conf="http://celtix.objectweb.org/transports/http/configuration">

  <bean id="celtix.{http://objectweb.org/hello_world_soap_http}SOAPService.
                                                        SoapPort.http-client"
        class="org.objectweb.celtix.bus.transports.http.http_client_config.
                                                    spring.HttpClientConfigBean">

      <property name="httpClient">
          <value>
              <http-conf:client ProxyServer="localhost" ProxyServerPort="5049"
                                AutoRedirect="true"/>
          </value>
      </property>

  </bean>
</beans>
```

An Example Application

The Celtix product includes several sample applications that illustrate configuration techniques. However, the best way to learn is to try it yourself, so here is a simple example. Besides giving you a chance to write a Celtix configuration file, this example will show you how to direct your application to actually use the configuration file.

The Application Code

For this example, you will build on the hello_world product demo. Build this demo and confirm that it runs successfully using both the ant utility and the java executable.

In this demo, the server mainline code sets the URL on which the application will listen for incoming requests.

```
package demo.hw.server;
import javax.xml.ws.Endpoint;

public class Server {

    protected Server() throws Exception {
        System.out.println("Starting Server");

        Object implementor = new GreeterImpl();
        String address = "http://localhost:9000/SoapContext/SoapPort";
        Endpoint.publish(address, implementor);
    }

    public static void main(String args[]) throws Exception {
        new Server();
        System.out.println("Server ready...");

        Thread.sleep(5 * 60 * 1000);
        System.out.println("Server exiting");
```

```
        System.exit(0);
    }
}
```

The client, however, obtains the URL from the WSDL file.

```
    <wsdl:service name="SOAPService">
        <wsdl:port binding="tns:Greeter_SOAPBinding" name="SoapPort">
            <soap:address location="http://localhost:9000/SoapContext/SoapPort"/>
        </wsdl:port>
    </wsdl:service>
```

In a text editor, open the WSDL file and change the TCP/IP port (any value is acceptable, just be certain that it is an unused port number). Save the file.

```
            <soap:address location="http://localhost:9002/SoapContext/SoapPort"/>
```

Now if you try to run the client, it will be unable to contact the server and the invocation requests will fail.

The Configuration File

Write a configuration file that sets the address property of the PortConfigBean. (NOTE: Remove the line break that follows **=**. This was added only for legibility in this document.)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans SYSTEM
     "http://celtix.objectweb.org/configuration/spring/celtix-spring-beans.dtd">

<beans
    xmlns:ct="http://celtix.objectweb.org/configuration/types">

  <bean id=
        "celtix.{http://objectweb.org/hello_world_soap_http}SOAPService.SoapPort"
        class="org.objectweb.celtix.bus.jaxws.port_config.spring.PortConfigBean">

      <property name="address">
          <value>
              <ct:stringValue>
                http://localhost:9000/SoapContext/SoapPort
              </ct:stringValue>
          </value>
      </property>
  </bean>

</beans>
```

Alternatively, you may code the **value** element as:

```
          <value>
                http://localhost:9000/SoapContext/SoapPort
          </value>
```

Next, add a **bean** element for a HttpClientConfigBean and include a **property** element to set authorization details and another **property** element to set the attributes related to the proxy server.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans SYSTEM
     "http://celtix.objectweb.org/configuration/spring/celtix-spring-beans.dtd">

<beans
    xmlns:ct="http://celtix.objectweb.org/configuration/types"
    xmlns:sec="http://celtix.objectweb.org/bus/configuration/security"
    xmlns:http-conf="http://celtix.objectweb.org/transports/http/configuration">

  <bean id=
        "celtix.{http://objectweb.org/hello_world_soap_http}SOAPService.SoapPort"
        class="org.objectweb.celtix.bus.jaxws.port_config.spring.PortConfigBean">

      <property name="address">
          <value>
              <ct:stringValue>
                http://localhost:9000/SoapContext/SoapPort
              </ct:stringValue>
```

```
        </value>
    </property>
  </bean>

  <bean id="celtix.{http://objectweb.orb/hello_world_soap_http}SOAPService.
                                                          SoapPort.http-client"
      class="org.objectweb.celtix.bus.transports.http.http_client_config.
                                                  spring.HttpClientConfigBean">

    <property name="authorization">
        <value>
            <sec:authorization>
                <sec:UserName>User</sec:UserName>
                <sec:Password>celtix</sec:Password>
            </sec:authorization>
        </value>
    </property>

    <!-- If you have a proxy server, remove comments and edit the values of
         ProxyServer and ProxyServerPort accordingly. -->
   <!--
    <property name="httpClient">
        <value>
            <http-conf:client ProxyServer="localhost" ProxyServerPort="5049"
                              AutoRedirect="true"/>
        </value>
    </property>
   -->

  </bean>
</beans>
```

The configuration file now includes two **bean** elements; the first represents configurable settings described in the **port-config.xml** metadata file, and the second represents configurable settings in the **http-client-config.xml** metadata file.

Save this file in text format into the directory ***<installationDirectory>*/celtix/samples/hello_world**; you may give the file any name; the next section assumes that the file is saved as **client.xml**. The WSDL file now has an incorrect URL while the configuration file and server mainline have the same URL.

Running the Example

Build the applications by issuing the **ant build** command. Then, using either the ant utility or the java executable, start the server application as described in the README.

When running the client application, you must force it to read the configuration file. You do this by providing a **-D** command line argument to the java executable.

To run the client using the java executable, enter the command:

```
java -Djava.util.logging.config.file=%CELTIX_HOME%\etc\logging.properties
     -Dceltix.config.file=file:///%CELTIX_HOME%\samples\hello_world\client.xml
     demo.hw.client.Client .\wsdl\hello_world.wsdl
```

To run the client using the ant utility, you will need to edit the **build.xml** file that is in the directory ***<installationDirectory>*/celtix/samples/hello_world**. Open this file in a text editor and modify the client target.

```
   <target name="client" description="run demo client">
       <property name="param" value=""/>
       <celtixrun classname="demo.hw.client.Client"
                  jvmarg1="-Dceltix.config.file=file:///${basedir}/client.xml"
                  param1="${basedir}/wsdl/hello_world.wsdl"
                  param2="${op}" param3="${param}"/>
   </target>
```

Appendix

The Schema Files

resources/schemas/configuration/std-types.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:tns="http://celtix.objectweb.org/configuration/types"
           targetNamespace="http://celtix.objectweb.org/configuration/types"
           elementFormDefault="qualified">

      <!--
        global elements
            - can be referenced from other parts of this schema or from other
            - schema documents; can appear as root element in valid documents
      -->

      <xs:element name="booleanValue" type="xs:boolean"/>
      <xs:element name="shortValue" type="xs:short"/>
      <xs:element name="intValue" type="xs:int"/>
      <xs:element name="integerValue" type="xs:integer"/>
      <xs:element name="longValue" type="xs:long"/>
      <xs:element name="floatValue" type="xs:float"/>
      <xs:element name="doubleValue" type="xs:double"/>
      <xs:element name="stringValue" type="xs:string"/>

      <xs:element name="unsignedShortValue" type="xs:unsignedShort"/>
      <xs:element name="unsignedIntValue" type="xs:unsignedInt"/>
      <xs:element name="unsignedLongValue" type="xs:unsignedLong"/>

      <xs:element name="stringListValue" type="tns:stringListType"/>
      <xs:element name="classNamespaceMappingListValue"
                  type="tns:classNamespaceMappingListType"/>

       <xs:complexType name="stringListType">
         <xs:sequence>
           <xs:element name="item" type="xs:string"
                       minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
       </xs:complexType>

       <xs:complexType name="classNamespaceMappingType">
         <xs:sequence>
           <xs:element name="classname" type="xs:string"/>
           <xs:sequence>
             <xs:element name="namespace" type="xs:anyURI" maxOccurs="unbounded"/>
           </xs:sequence>
         </xs:sequence>
       </xs:complexType>

       <xs:complexType name="classNamespaceMappingListType">
         <xs:sequence id="mappings">
           <xs:element name="map" type="tns:classNamespaceMappingType"
                       minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
       </xs:complexType>
</xs:schema>
```

resources/schemas/configuration/metadata.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:tns="http://celtix.objectweb.org/configuration/metadata"
           targetNamespace="http://celtix.objectweb.org/configuration/metadata"
           elementFormDefault="qualified" attributeFormDefault="unqualified">
```

```
<xs:element name="config" type="tns:configType">
  <xs:annotation>
    <xs:documentation>
        Describes the complete set of configuration
        items in a component
    </xs:documentation>
  </xs:annotation>
  <xs:unique name="nameConstraint">
    <xs:selector xpath="./tns:configItem/tns:name"/>
    <xs:field xpath="."/>
  </xs:unique>
</xs:element>

<xs:complexType name="configType">
  <xs:sequence>
    <xs:element name="configImport" type="tns:importType"
                minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="configItem" type="tns:configItemType"
                minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="namespace" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:complexType name="importType">
  <xs:attribute name="namespace" type="xs:anyURI" use="required"/>
  <xs:attribute name="location" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:complexType name="configItemType">
  <xs:annotation>
    <xs:documentation>
        Describes an individual configuration item
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="name" type="tns:nameType"
                minOccurs="1" maxOccurs="1"/>
    <xs:element name="type" type="tns:typeType"
                minOccurs="1" maxOccurs="1" />
    <xs:element name="description" type="tns:descriptionType"
                nillable="true" minOccurs="0"/>
    <xs:element name="lifecyclePolicy" type="tns:lifecyclePolicyType"
                nillable="false" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="lifecyclePolicyType">
  <xs:annotation>
    <xs:documentation>
        Specifies if and when the value of a configuration item can be
        modified
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="static"/>
    <xs:enumeration value="process"/>
    <xs:enumeration value="bus"/>
    <xs:enumeration value="dynamic"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="nameType">
  <xs:restriction base="xs:string">
    <xs:pattern value="\S+"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="typeType">
  <xs:restriction base="xs:string">
    <xs:pattern value="\S+"/>
```

```
        </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="descriptionType">
        <xs:restriction base="xs:string">
          <xs:whiteSpace value="preserve"/>
        </xs:restriction>
      </xs:simpleType>
</xs:schema>
```

resources/schemas/configuration/jaxws-types.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:tns="http://celtix.objectweb.org/bus/jaxws/configuration/types"
      targetNamespace="http://celtix.objectweb.org/bus/jaxws/configuration/types"
      elementFormDefault="qualified">

      <!--
        global elements
            - can be referenced from other parts of this schema or from other
            - schema documents; can appear as root element in valid documents
      -->
      <xs:element name="handlerChainValue" type="tns:handlerChainType"/>

      <!--
        type definitions
      -->
      <xs:complexType name="handlerType">
        <xs:sequence>
          <xs:element name="name" type="xs:string" minOccurs="0"/>
          <xs:element name="className" type="xs:string"/>
          <xs:element name="init-param" type="tns:handlerInitParamType"
                      nillable="false" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>

      <xs:complexType name="handlerChainType">
        <xs:sequence>
          <xs:element name="handler" type="tns:handlerType"
                      minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>

      <xs:complexType name="handlerInitParamType">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:element name="param-name" nillable="false"/>
          <xs:element name="param-value"/>
        </xs:sequence>
      </xs:complexType>
</xs:schema>
```

resources/schemas/configuration/security.xsd

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

<xs:schema
targetNamespace="http://celtix.objectweb.org/bus/configuration/security"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
           xmlns:tns="http://celtix.objectweb.org/bus/configuration/security"
           jaxb:version="2.0">

    <xs:annotation>
        <xs:appinfo>
            <jaxb:globalBindings generateIsSetMethod="true"/>
        </xs:appinfo>
```

```
    </xs:annotation>

    <xs:complexType name="AuthorizationPolicy">
        <xs:annotation>
            <xs:documentation>
            Policies for controlling basic authentication
            </xs:documentation>
        </xs:annotation>

        <xs:sequence>
            <xs:element name="UserName" type="xs:string" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>
                    User Name for  server BASIC authentication login
                    (some servers require users to authenticate with the server
                    -- see also Password, AuthorizationType, and Authorization)
                    </xs:documentation>
                </xs:annotation>
            </xs:element>

            <xs:element name="Password" type="xs:string" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>
                    Password for server BASIC authentication login
                    (some servers require users to authenticate with the server
                    -- see also UserName, AuthorizationType, and Authorization)
                    </xs:documentation>
                </xs:annotation>
            </xs:element>

            <xs:element name="AuthorizationType" type="xs:string" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>
                    Type of authentication to use with server, if not using
                    username and password based "BASIC" authentication.
                    If username and password are used, this does not need to
                    be set.
                    Some servers require users to authenticate with the server
                    -- see also UserName, Password, and Authorization.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>

            <xs:element name="Authorization" type="xs:string" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>
                    Actual authentication data for server, if not using
                    username and password based "BASIC" authentication
                    If username and password are used, this does not need to
                    be set.
                    Some servers require users to authenticate with the server
                    -- see also UserName, Password, and AuthorizationType.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="authorization" type="tns:AuthorizationPolicy"/>

    <xs:complexType name="SSLPolicy">
        <xs:annotation>
            <xs:documentation>
            Policies for controlling SSL encryption
            </xs:documentation>
        </xs:annotation>

       <xs:sequence>
             <!--SSL related elements-->
            <xs:element name="UseSecureSockets" type="xs:boolean" minOccurs="0"
                        default="false">
```

```
            <xs:annotation>
                <xs:documentation>
                If true, SSL or TLS connection will be expected by the
                server.  SSL and TLS are public standards, and when they
                are used,  the connection is often referred to
                as HTTPS instead of HTTP.
                </xs:documentation>
            </xs:annotation>
        </xs:element>

        <xs:element name="Certificate" type="xs:string" minOccurs="0">
            <xs:annotation>
                <xs:documentation>
                Some servers may require the client to also present a
                certificate for authentication of the client by the server.
                This field should contain the PKCS12 encoded certificate
                issued by the certificate authority.
                </xs:documentation>
            </xs:annotation>
        </xs:element>

        <xs:element name="CertificateChain" type="xs:string" minOccurs="0">
            <xs:annotation>
                <xs:documentation>
                Certificates can be issued by intermediate certificate
                authorities that are not trusted by the Server,
                but have had their certificates issues by a trusted
                authority.  This option allows the certificate chain of PEM
                encoded X509 certificates to be presented for verification.
                Optional.
                This will be deprecated due to the move to PKCS12.
                </xs:documentation>
            </xs:annotation>
        </xs:element>

        <xs:element name="PrivateKey" type="xs:string" minOccurs="0">
            <xs:annotation>
                <xs:documentation>
                The PEM encoded private key corresponding to X509 certificate
                listed in  ClientCertificate.  Required if ClientCertificate
                is specified, and cannot be  specified if ClientCertificate
                is not specified.
                This will be deprecated due to the move to PKCS12.
                </xs:documentation>
            </xs:annotation>
        </xs:element>

        <xs:element name="PrivateKeyPassword" type="xs:string" minOccurs="0">
            <xs:annotation>
                <xs:documentation>
                If the PEM encoded ClientPrivateKey field is encrypted with a
                password, this is the password used to decrypt it.
                These keys are typically encrypted by the
                certificate authority when sending the key over a public
                network, and password is delivered by a secure means.
                </xs:documentation>
            </xs:annotation>
        </xs:element>

        <xs:element name="TrustedRootCertificates" type="xs:string"
                    minOccurs="0">
            <xs:annotation>
                <xs:documentation>
                The client should specify a list of PEM encoded X509
                certificates for certificate authorities that are trusted.
                This will be used to validate the certificate presented
                by the server.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
```

```
    </xs:complexType>

    <xs:element name="ssl" type="tns:SSLPolicy"/>
</xs:schema>
```

The XML Metadata Files

resources/config-metadata/bus-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cm:config
    xmlns:cm="http://celtix.objectweb.org/configuration/metadata"
    xmlns:ct="http://celtix.objectweb.org/configuration/types"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://celtix.objectweb.org/configuration/metadata
    http://celtix.objectweb.org/configuration/metadata.xsd"
    namespace="http://celtix.objectweb.org/bus/bus-config">

   <!-- configuration metadata for component Bus  -->

    <cm:configImport location="schemas/configuration/std-types.xsd"
                     namespace="http://celtix.objectweb.org/configuration/types"/>

    <cm:configItem>
        <cm:name>bindingFactories</cm:name>
        <cm:type>ct:classNamespaceMappingListType</cm:type>
        <cm:description>
        Specifies (by classname) the binding factories to be registered with the
        bus, and, for each factory, the list of supported binding URIs.
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <ct:classNamespaceMappingListValue>
            <ct:map>
                <ct:classname>
                    org.objectweb.celtix.bus.bindings.soap.SOAPBindingFactory
                </ct:classname>
                <ct:namespace>
                    http://schemas.xmlsoap.org/wsdl/soap/
                </ct:namespace>
                <ct:namespace>
                    http://schemas.xmlsoap.org/wsdl/soap/http
                </ct:namespace>
                <ct:namespace>
                    http://celtix.objectweb.org/transports/jms
                </ct:namespace>
            </ct:map>
        </ct:classNamespaceMappingListValue>
    </cm:configItem>

    <cm:configItem>
        <cm:name>transportFactories</cm:name>
        <cm:type>ct:classNamespaceMappingListType</cm:type>
        <cm:description>
        Specifies (by classname) the transport factories to be registered
        with the bus, and, for each factory, the list of supported
        transport URIs.
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <ct:classNamespaceMappingListValue>
            <ct:map>
                <ct:classname>
                    org.objectweb.celtix.bus.transports.http.HTTPTransportFactory
                </ct:classname>
                <ct:namespace>
                    http://schemas.xmlsoap.org/wsdl/soap/
                </ct:namespace>
                <ct:namespace>
                    http://schemas.xmlsoap.org/wsdl/soap/http
                </ct:namespace>
```

```
                <ct:namespace>
                    http://celtix.objectweb.org/transports/http/configuration
                </ct:namespace>
            </ct:map>
            <ct:map>
                <ct:classname>
                    org.objectweb.celtix.bus.transports.jms.JMSTransportFactory
                </ct:classname>
                <ct:namespace>
                    http://celtix.objectweb.org/transports/jms
                </ct:namespace>
                <ct:namespace>
                    http://celtix.objectweb.org/transports/jms/configuration
                </ct:namespace>
            </ct:map>
        </ct:classNamespaceMappingListValue>
    </cm:configItem>

</cm:config>
```

resources/config-metadata/endpoint-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cm:config
    xmlns:cm="http://celtix.objectweb.org/configuration/metadata"
    xmlns:std-types="http://celtix.objectweb.org/configuration/types"
    xmlns:jaxws-types="http://celtix.objectweb.org/bus/jaxws/configuration/types"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
        "http://celtix.objectweb.org/configuration/metadata metadata.xsd"
    namespace="http://celtix.objectweb.org/bus/jaxws/endpoint-config">

   <!-- configuration metadata for component Endpoint  -->

    <cm:configImport location="schemas/configuration/std-types.xsd"
                     namespace=
                    "http://celtix.objectweb.org/configuration/types"/>
    <cm:configImport location="schemas/configuration/jaxws-types.xsd"
                    namespace=
                    "http://celtix.objectweb.org/bus/jaxws/configuration/types"/>

    <cm:configItem>
        <cm:name>handlerChain</cm:name>
        <cm:type>jaxws-types:handlerChainType</cm:type>
        <cm:description>
        Specifies the list of handlers to be used on an endpoint. The list can
        contain a mix of logical and protocol handlers.
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
    </cm:configItem>
</cm:config>
```

resources/config-metadata/service-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cm:config
    xmlns:cm="http://celtix.objectweb.org/configuration/metadata"
    xmlns:std-types="http://celtix.objectweb.org/configuration/types"
    xmlns:jaxws-types="http://celtix.objectweb.org/bus/jaxws/configuration/types"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
        "http://celtix.objectweb.org/configuration/metadata metadata.xsd"
    namespace="http://celtix.objectweb.org/bus/jaxws/service-config">

   <!-- configuration metadata for component Service  -->

    <cm:configImport location="schemas/configuration/std-types.xsd"
                     namespace=
                    "http://celtix.objectweb.org/configuration/types"/>
    <cm:configImport location="schemas/configuration/jaxws-types.xsd"
```

```
                    namespace=
                   "http://celtix.objectweb.org/bus/jaxws/configuration/types"/>

    <cm:configItem>
        <cm:name>handlerChain</cm:name>
        <cm:type>jaxws-types:handlerChainType</cm:type>
        <cm:description>
        Specifies the list of handlers to be used for a service.
        The list can contain a mix of logical and protocol handlers.
        It is up to the HandlerRegister
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
    </cm:configItem>
</cm:config>
```

resources/config-metadata/port-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cm:config
    xmlns:cm="http://celtix.objectweb.org/configuration/metadata"
    xmlns:ct="http://celtix.objectweb.org/configuration/types"
    xmlns:jaxws-types="http://celtix.objectweb.org/bus/jaxws/configuration/types"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
        "http://celtix.objectweb.org/configuration/metadata metadata.xsd"
    namespace="http://celtix.objectweb.org/bus/jaxws/port-config">

   <!-- configuration metadata for component Port  -->

    <cm:configImport location="schemas/configuration/std-types.xsd"
                     namespace="http://celtix.objectweb.org/configuration/types"/>

    <cm:configItem>
        <cm:name>address</cm:name>
        <cm:type>xs:string</cm:type>
        <cm:description>
        Address to which the service should connect.
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- no default -->
    </cm:configItem>

    <cm:configItem>
        <cm:name>bindingId</cm:name>
        <cm:type>xs:string</cm:type>
        <cm:description>
        Binding to be used by the service.
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- no default -->
    </cm:configItem>

    <cm:configItem>
        <cm:name>transportId</cm:name>
        <cm:type>xs:string</cm:type>
        <cm:description>
        Transport to be used by the service. If not specified,
        the transport will be determined by the form of the address.
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- no default -->
    </cm:configItem>
</cm:config>
```

resources/config-metadaata/http-client-config.xml.

This file is subject to revisions in future releases of Celtix.

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

```
<cm:config
    xmlns:cm="http://celtix.objectweb.org/configuration/metadata"
    xmlns:http-conf="http://celtix.objectweb.org/transports/http/configuration"
    xmlns:sec="http://celtix.objectweb.org/bus/configuration/security"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    namespace=
        "http://celtix.objectweb.org/bus/transports/http/http-client-config">

    <cm:configImport namespace=
                     "http://celtix.objectweb.org/transports/http/configuration"
                     location="schemas/wsdl/http-conf.xsd"/>
    <cm:configImport namespace=
                     "http://celtix.objectweb.org/bus/configuration/security"
                     location="schemas/configuration/security.xsd"/>

    <cm:configItem>
        <cm:name>httpClient</cm:name>
        <cm:type>http-conf:HTTPClientPolicy</cm:type>
        <cm:description>HTTP client configuration properties.
            Used for configuring a HTTP client port.
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- attribute values are defaulted -->
        <http-conf:client></http-conf:client>
    </cm:configItem>

    <cm:configItem>
        <cm:name>authorization</cm:name>
        <cm:type>sec:AuthorizationPolicy</cm:type>
        <cm:description>
            Policies for controlling basic authentication
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
    </cm:configItem>

    <cm:configItem>
        <cm:name>proxyAuthorization</cm:name>
        <cm:type>sec:AuthorizationPolicy</cm:type>
        <cm:description>
            Policies for controlling authentication to a proxy server
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
    </cm:configItem>

    <cm:configItem>
        <cm:name>ssl</cm:name>
        <cm:type>sec:SSLPolicy</cm:type>
        <cm:description>
            Policies for controlling SSL encryption
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- attribute values are defaulted -->
        <sec:ssl></sec:ssl>
    </cm:configItem>
</cm:config>
```

resources/config-metadata/http-server-config.xml

This file is subject to revisions in future releases of Celtix.

```
<?xml version="1.0" encoding="UTF-8"?>
<cm:config
    xmlns:cm="http://celtix.objectweb.org/configuration/metadata"
    xmlns:http-conf="http://celtix.objectweb.org/transports/http/configuration"
    xmlns:sec="http://celtix.objectweb.org/bus/configuration/security"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    namespace=
        "http://celtix.objectweb.org/bus/transports/http/http-server-config">
```

```
    <cm:configImport namespace=
                     "http://celtix.objectweb.org/transports/http/configuration"
                     location="schemas/wsdl/http-conf.xsd"/>
    <cm:configImport namespace=
                     "http://celtix.objectweb.org/bus/configuration/security"
                     location="schemas/configuration/security.xsd"/>

    <cm:configItem>
        <cm:name>httpServer</cm:name>
        <cm:type>http-conf:HTTPServerPolicy</cm:type>
        <cm:description>HTTP server configuration properties.
            Used for configuring a HTTP server port.
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- attribute values are defaulted -->
        <http-conf:server></http-conf:server>
    </cm:configItem>

    <cm:configItem>
        <cm:name>authorization</cm:name>
        <cm:type>sec:AuthorizationPolicy</cm:type>
        <cm:description>
            Policies for controlling basic authentication
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- no default -->
    </cm:configItem>

    <cm:configItem>
        <cm:name>ssl</cm:name>
        <cm:type>sec:SSLPolicy</cm:type>
        <cm:description>
            Policies for controlling SSL encryption
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- attribute values are defaulted -->
        <sec:ssl></sec:ssl>
    </cm:configItem>
</cm:config>
```

resources/config-metadata/http-listener-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cm:config
    xmlns:cm="http://celtix.objectweb.org/configuration/metadata"
    xmlns:http-conf="http://celtix.objectweb.org/transports/http/configuration"
    xmlns:sec="http://celtix.objectweb.org/bus/configuration/security"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    namespace=
        "http://celtix.objectweb.org/bus/transports/http/http-listener-config">

    <cm:configImport namespace=
                     "http://celtix.objectweb.org/transports/http/configuration"
                     location="schemas/wsdl/http-conf.xsd"/>
    <cm:configImport namespace=
                     "http://celtix.objectweb.org/bus/configuration/security"
                     location="schemas/configuration/security.xsd"/>

    <cm:configItem>
        <cm:name>httpListener</cm:name>
        <cm:type>http-conf:HTTPListenerPolicy</cm:type>
        <cm:description>
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
        <!-- attribute values are defaulted -->
        <http-conf:listener></http-conf:listener>
    </cm:configItem>
    <cm:configItem>
```

```
        <cm:name>ssl</cm:name>
        <cm:type>sec:SSLPolicy</cm:type>
        <cm:description>
            Policies for controlling SSL encryption
        </cm:description>
        <cm:lifecyclePolicy>bus</cm:lifecyclePolicy>
    </cm:configItem>
</cm:config>
```