Server HTTP Transport

Server HTTP Transport

HTTP server endpoints can specify a number of HTTP connection attributes including if it will honor keep alive requests, how it interacts with caches, and how tolerant it is of errors in communicating with a consumer.

A server endpoint can be configured using two mechanisms:

- Configuration
- WSDL

Using Configuration

Namespace

The elements used to configure an HTTP provider endpoint are defined in the namespace `http://cxf.apache.org/transports/http/configuration`. It is commonly referred to using the prefix `http-conf`. In order to use the HTTP configuration elements you will need to add the lines shown below to the beans element of your endpoint's configuration file. In addition, you will need to add the configuration elements' namespace to the `xsi:schemaLocation` attribute.

Adding the Configuration Namespace

```
<beans ...
       xmlns:http-conf="http://cxf.apache.org/transports/http/configuration
       ...
       xsi:schemaLocation="...
          http://cxf.apache.org/transports/http/configuration
          http://cxf.apache.org/schemas/configuration/http-conf.xsd
       ...">
```

The `destination` element

You configure an HTTP server endpoint using the `http-conf:destination` element and its children. The `http-conf:destination` element takes a single attribute, `name`, the specifies the WSDL port element that corresponds to the endpoint. The value for the `name` attribute takes the form *portQName*`.http-destination`. The example below shows the `http-conf:destination` element that would be used to add configuration for an endpoint that was specified by the WSDL fragment `<port binding="widgetSOAPBinding" name="widgetSOAPPort>` if the endpoint's target namespace was `http://widgets.widgetvendor.net`.

http-conf:destination Element

```
...
  <http-conf:destination name="{http://widgets/widgetvendor.net}widgetSOAPPort.http-destination">
    ...
  </http-conf:destination>
...
```

The `http-conf:destination` element has a number of child elements that specify configuration information. They are described below.

Element	Description
`http-conf:server`	Specifies the HTTP connection properties.
`http-conf:contextMatchStrategy`	Specifies the parameters that configure the context match strategy for processing HTTP requests.
`http-conf:fixedParameterOrder`	Specifies whether the parameter order of an HTTP request handled by this destination is fixed.

The `server` element

The `http-conf:server` element is used to configure the properties of a server's HTTP connection. Its attributes, described below, specify the connection's properties.

Attribute	Description

`ReceiveTimeout`	Sets the length of time, in milliseconds, the server tries to receive a request before the connection times out. The default is 30000. Use 0 to specify that the server will not timeout.
`SuppressClientSendErrors`	Specifies whether exceptions are to be thrown when an error is encountered on receiving a request. The default is `false`; exceptions are thrown on encountering errors.
`SuppressClientReceiveErrors`	Specifies whether exceptions are to be thrown when an error is encountered on sending a response to a client. The default is `false`; exceptions are thrown on encountering errors.
`HonorKeepAlive`	Specifies whether the server honors requests for a connection to remain open after a response has been sent. The default is `true`; keep-alive requests are honored.
`RedirectURL`	Specifies the URL to which the client request should be redirected if the URL specified in the client request is no longer appropriate for the requested resource. In this case, if a status code is not automatically set in the first line of the server response, the status code is set to 302 and the status description is set to Object Moved. The value is used as the value of the HTTP `RedirectURL` property.
`CacheControl`	Specifies directives about the behavior that must be adhered to by caches involved in the chain comprising a response from a server to a client.
`ContentLocation`	Sets the URL where the resource being sent in a response is located.
`ContentType`	Specifies the media type of the information being sent in a response. Media types are specified using multipurpose internet mail extensions (MIME) types. The value is used as the value of the HTTP `ContentType` location.
`ContentEncoding`	Specifies any additional content encodings that have been applied to the information being sent by the service provider. Content encoding labels are regulated by the Internet Assigned Numbers Authority (IANA). Possible content encoding values include zip, gzip, compress, deflate, and identity. This value is used as the value of the HTTP `ContentEncoding` property.
`ServerType`	Specifies what type of server is sending the response. Values take the form program-name/version. For example, Apache/1.2.5.

Example

The example below shows a the configuration for an HTTP service provider endpoint that honors keep alive requests and suppresses all communication errors.

HTTP Service Provider Endpoint Configuration

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"
       xsi:schemaLocation="http://cxf.apache.org/transports/http/configuration
           http://cxf.apache.org/schemas/configuration/http-conf.xsd
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd">

  <http-conf:destination
     name="{http://apache.org/hello_world_soap_http}SoapPort.http-destination">
    <http-conf:server SuppressClientSendErrors="true"
                      SuppressClientReceiveErrors="true"
                      HonorKeepAlive="true" />
  </http-conf:destination>
</beans>
```

Using WSDL

Namespace

The WSDL extension elements used to configure an HTTP server endpoint are defined in the namespace `http://cxf.apache.org/transports/http/configuration`. It is commonly refered to using the prefix `http-conf`. In order to use the HTTP configuration elements you will need to add the line shown below to the `definitions` element of your endpoint's WSDL document.

HTTP Provider WSDL Element's Namespace

```
<definitions ...
       xmlns:http-conf="http://cxf.apache.org/transports/http/configuration
```

The `server` element

The `http-conf:server` element is used to specify the connection properties of an HTTP server in a WSDL document. The `http-conf:server` element is a child of the WSDL port element. It has the same attributes as the `server` element used in the configuration file.

Example

The example below shows a WSDL fragment that configures an HTTP server endpoint to specify that it will not interact with caches.

WSDL to Configure an HTTP Service Provider Endpoint

```
<service ...>
  <port ...>
    <soap:address ... />
    <http-conf:server CacheControl="no-cache" />
  </port>
</service>
```

Server Cache Control Directives

The table below lists the cache control directives supported by an HTTP server.

Directive	Behavior
`no-cache`	Caches cannot use a particular response to satisfy subsequent requests without first revalidating that response with the server. If specific response header fields are specified with this value, the restriction applies only to those header fields within the response. If no response header fields are specified, the restriction applies to the entire response.
`public`	Any cache can store the response.
`private`	Public (shared) caches cannot store the response because the response is intended for a single user. If specific response header fields are specified with this value, the restriction applies only to those header fields within the response. If no response header fields are specified, the restriction applies to the entire response.
`no-store`	Caches must not store any part of response or any part of the request that invoked it.
`no-transform`	Caches must not modify the media type or location of the content in a response between a server and a client.
`must-revalidate`	Caches must revaildate expired entries that relate to a response before that entry can be used in a subsequent response.
`proxy-revalidate`	Means the same as must-revalidate, except that it can only be enforced on shared caches and is ignored by private unshared caches. If using this directive, the public cache directive must also be used.
`max-age`	Clients can accept a response whose age is no greater that the specified number of seconds.
`s-max-age`	Means the same as max-age, except that it can only be enforced on shared caches and is ignored by private unshared caches. The age specified by s-max-age overrides the age specified by max-age. If using this directive, the proxy-revalidate directive must also be used.
`cache-extension`	Specifies additional extensions to the other cache directives. Extensions might be informational or behavioral. An extended directive is specified in the context of a standard directive, so that applications not understanding the extended directive can at least adhere to the behavior mandated by the standard directive.