

# Features

## Features in CXF

A Feature in CXF is a way of adding capabilities to a Server, Client or Bus. For example, you could add the ability to log messages for each of these objects, by configuring them with a `LoggingFeature`. To implement a Feature, you must subclass [AbstractFeature](#) below. By default the initialize methods all delegate to `initializeProvider(InterceptorProvider)`, so if you're simply adding interceptors to a Server, Client, or Bus, this allows you to add them easily.

```
public abstract class AbstractFeature extends WebServiceFeature implements Feature {
    public String getID() {
        return getClass().getName();
    }
    public void initialize(Server server, Bus bus) {
        initializeProvider(server.getEndpoint(), bus);
    }
    public void initialize(Client client, Bus bus) {
        initializeProvider(client, bus);
    }
    public void initialize(InterceptorProvider interceptorProvider, Bus bus) {
        initializeProvider(interceptorProvider, bus);
    }
    public void initialize(Bus bus) {
        initializeProvider(bus, bus);
    }
    protected void initializeProvider(InterceptorProvider provider, Bus bus) {
        // you could customize the interceptors in the provider here
    }
    /**
     * Convenience method to extract a feature by type from an active list.
     *
     * @param features the given feature list
     * @param type the feature type required
     * @return the feature of the specified type if active
     */
    public static <T> T getActive(List<? extends Feature> features,
                                Class<T> type) {
        T active = null;
        if (features != null) {
            for (Feature feature : features) {
                if (type.isInstance(feature)) {
                    active = type.cast(feature);
                    break;
                }
            }
        }
        return active;
    }
}
```

## Writing and configuring the Feature

CXF provides several features to configure commonly used capabilities, such as logging, failover, policies, addressing, and reliable messaging. You can go to the [FeaturesList](#) for more information.

## Writing a Feature

It is very easy to write a new feature; your feature just needs to extend the [AbstractFeature](#) and implement `initializeProvider` or write customizing code for configuring client or server interceptors. Here is an example for implementing the logging feature.

```

public class LoggingFeature extends AbstractFeature {
    private static final int DEFAULT_LIMIT = 100 * 1024;
    private static final LoggingInInterceptor IN = new LoggingInInterceptor(DEFAULT_LIMIT);
    private static final LoggingOutInterceptor OUT = new LoggingOutInterceptor(DEFAULT_LIMIT);

    int limit = DEFAULT_LIMIT;

    @Override
    protected void initializeProvider(InterceptorProvider provider, Bus bus) {
        if (limit == DEFAULT_LIMIT) {
            provider.getInInterceptors().add(IN);
            provider.getInFaultInterceptors().add(IN);
            provider.getOutInterceptors().add(OUT);
            provider.getOutFaultInterceptors().add(OUT);
        } else {
            LoggingInInterceptor in = new LoggingInInterceptor(limit);
            LoggingOutInterceptor out = new LoggingOutInterceptor(limit);
            provider.getInInterceptors().add(in);
            provider.getInFaultInterceptors().add(in);
            provider.getOutInterceptors().add(out);
            provider.getOutFaultInterceptors().add(out);
        }
    }

    /**
     * This function has no effect at this time.
     * @param lim
     */
    public void setLimit(int lim) {
        limit = lim;
    }

    /**
     * Retrieve the value set with {@link #setLimit(int)}.
     * @return
     */
    public int getLimit() {
        return limit;
    }
}

```

## Adding a Feature programmatically

To add the feature to both server and client, you can use the Feature annotation on the service class

```

@org.apache.cxf.feature.Features (features = "org.apache.cxf.jaxws.service.AnnotationFeature")
public class HelloServiceImpl implements HelloService {
    public String sayHi() {
        return "HI";
    }
}

```

You can also add the feature to the server by using `ServerFactoryBean`, or the client by using the `ClientFactoryBean` or the `ClientProxyFactoryBean`

```

import org.apache.cxf.frontend.ServerFactoryBean;
import org.apache.cxf.frontend.ClientFactoryBean;
import org.apache.cxf.frontend.ClientProxyFactoryBean;
...
ServerFactoryBean serverFactoryBean = new ServerFactoryBean();
MyFeature myFeature = new MyFeature();
// added my feature to the serverFactoryBean
serverFactoryBean.setFeatures(Collections.singletonList(myFeature));
...

ClientFactoryBean clientFactoryBean = new ClientFactoryBean();
clientFactoryBean.setFeatures(Collections.singletonList(myFeature));
...

ClientProxyFactoryBean clientFactoryBean = new ClientProxyFactoryBean();
clientFactoryBean.setFeatures(Collections.singletonList(policyFeature));

```

It is also possible to add a feature when publishing an Endpoint via `Endpoint.publish`. For example, the Logging feature can be added as follows:

#### Endpoint.publish

```

import javax.xml.ws.Endpoint;
import org.apache.cxf.ext.logging.LoggingFeature;
...
CustomerService implementor = new CustomerServiceImpl();
Endpoint.publish("http://localhost:9090/CustomerServicePort",
    implementor,
    new LoggingFeature());

```

## Adding a Feature through configuration

Here are some examples on using configuration files to add features. You can find more information about the CXF provides features at [FeaturesList](#).

```

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:cxf="http://cxf.apache.org/core"
       xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="
http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd">

    <!-- adding the feature to the bus-->
    <cxf:bus>
        <cxf:features>
            <cxf:logging/>
        </cxf:features>
    </cxf:bus>

    <bean id="myfeature" class="com.example.Myfeature"/>

    <!-- adding the feature to the client -->
    <jaxws:client id="client"
        serviceClass="org.apache.hello_world_soap_http.Greeter"
        wsdlLocation="wsdl/hello_world.wsdl">
        <jaxws:features>
            <bean ref="myfeature" />
        </jaxws:features>
    </jaxws:client>

    <!-- adding the feature to the server -->
    <jaxws:server id="server" serviceBean="org.apache.hello_world_soap_http.GreetImpl" address="
http://localhost:8080/simpleWithAddress">
        <jaxws:features>
            <bean class="com.example.Myfeature"/>
        </jaxws:features>
    </jaxws:server>

</beans>

```