OpenSplice | DDS

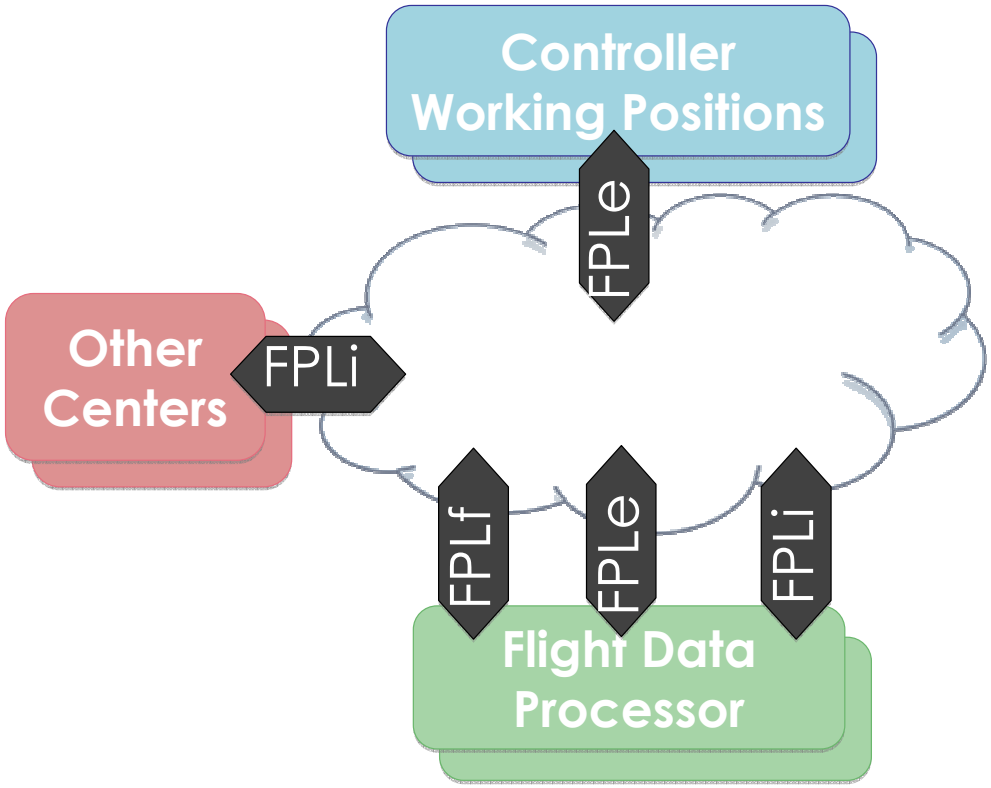
Integration Patterns for Mission Critical System of Systems

Julien ENOCH
Engineering Team Lead
PrismTech
julien.enoch@prismtech.com

Copyright PrismTech 2012

System Integration Challenges

- Subsystems integration
  - Complex distributed systems are often made up by different subsystems
  - Ideally: decoupled and independently evolvable
  - In fact:
    - mutual dependencies
    - coupling
    - different representation of the same information

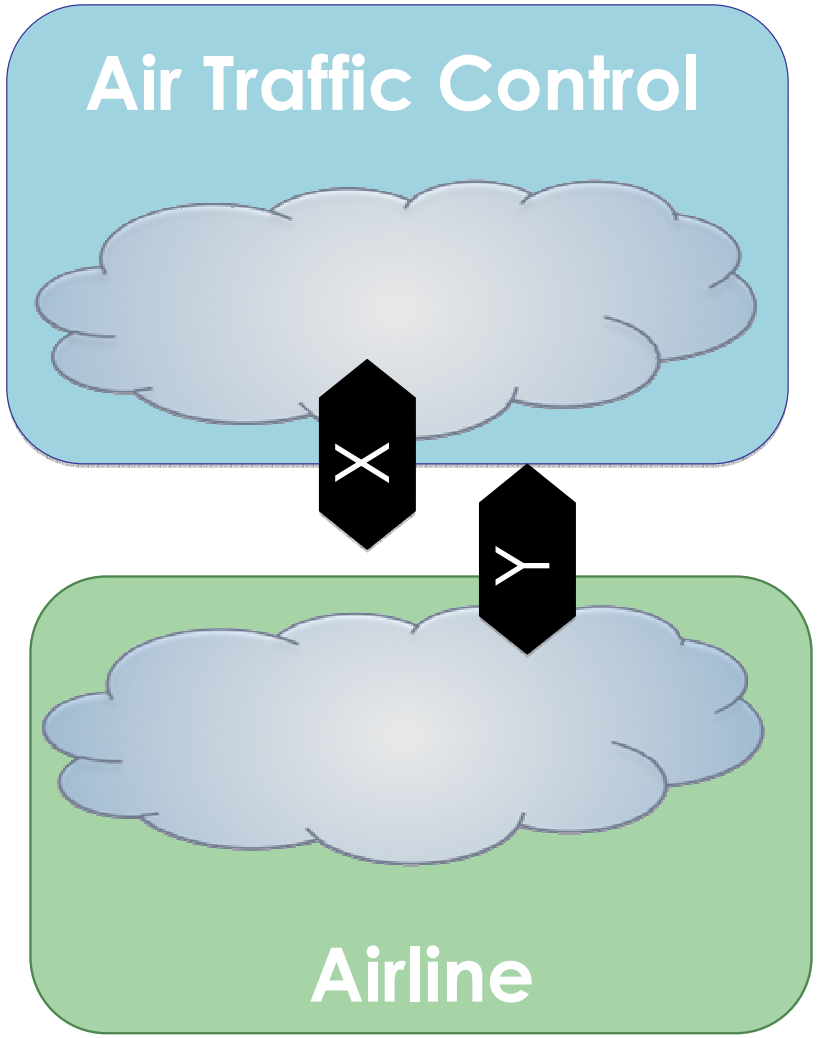


Copyright PrismTech 2012

OpenSplice DDS

System Integration Challenges

- System of Systems
  - Independently developed
    - different data model (e.g. Cartesian vs. Polar Coordinates)
    - different topic names
    - different domains
    - etc…
  - This differences need to be addressed in order for systems to be integrated



Copyright PrismTech 2012

System Integration Challenges

- Technologies Integration
  - Different systems (legacy…) based on different technologies (non-DDS)
  - Several systems need to make information available through a wide-set of "media"
  - Others need to inject data from various protocols (TCP, HTTP…)

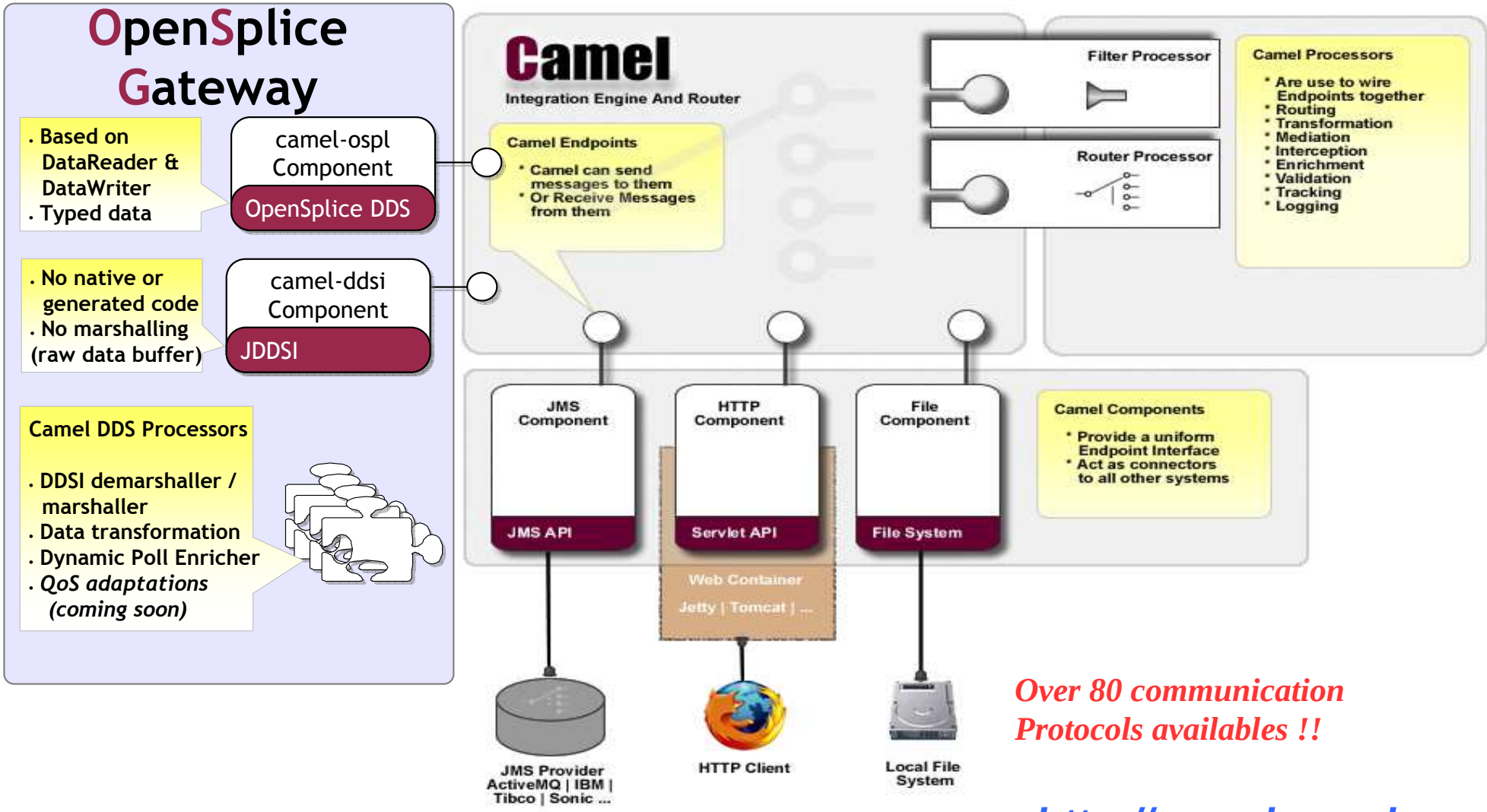
-WebSockets-

JMS

OpenSplice DDS

Copyright PrismTech 2012

OpenSplice Gateway



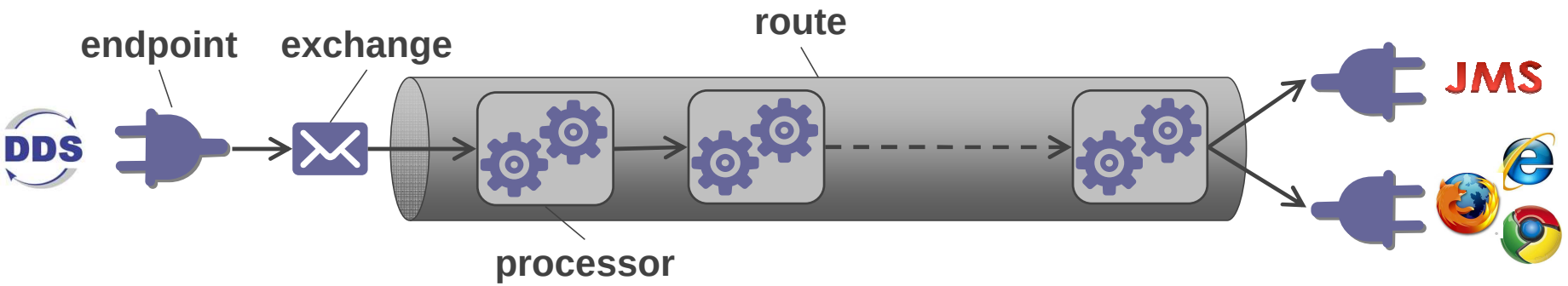
Over 80 communication Protocols availables !!

http://camel.apache.org

Copyright PrismTech 2012

Apache Camel

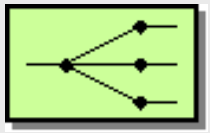
- Based on "Enterprise Integration Patterns" book
  - *by G. Hohpe, B. Woolf (ed. Addisson Wesley)*
- Routing and mediation engine
- User defines routes for messages:
  - from **IN endpoint** to **OUT endpoint(s)**
  - using patterns implemented as **Processors**

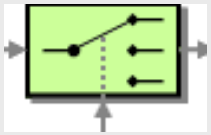


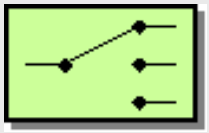
Copyright PrismTech 2012

Some Camel Patterns

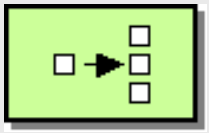
Message Routing

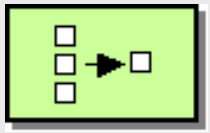

Recipient List

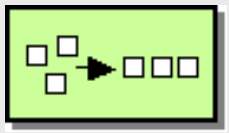

Dynamic Router

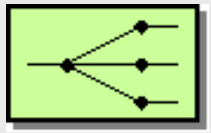

Content Based Router

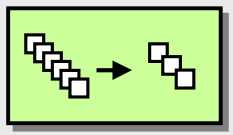

Message filter


Splitter

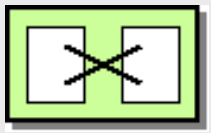

Aggregator

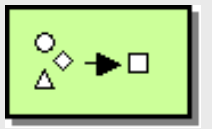

Resequencer

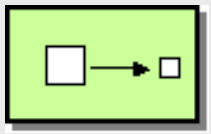

Load Balancer

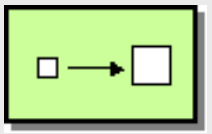

Throttler

Message Transformation

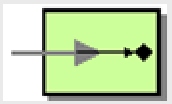

Message Translator

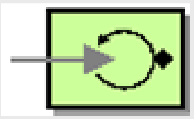

Normalizer

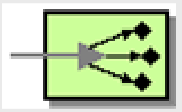

Content filter


Content enricher

Messaging Endpoints


Event Driven Consumer


Polling Consumer

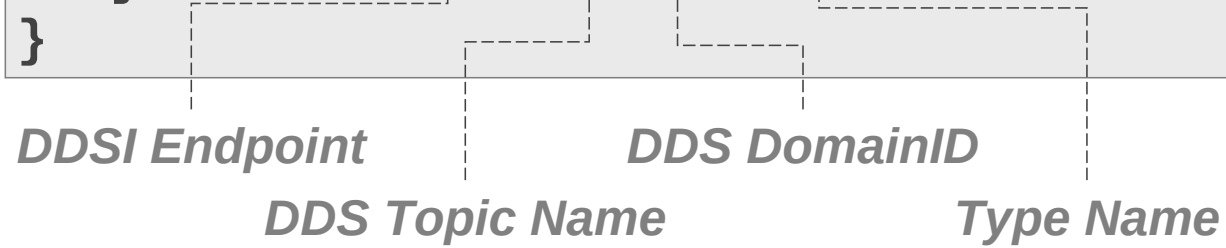

Competing Consumers

Copyright PrismTech 2012

Camel routes

- Endpoints are defined via **URIs**
  - Ex.: “http://localhost:8080” , “jms:topic:XYZ”, “tcp:localhost:88” …
- A route can be defined in **Spring** (XML), or using either the **Java** or **Scala** DSL
  - Ex. With Java DSL:

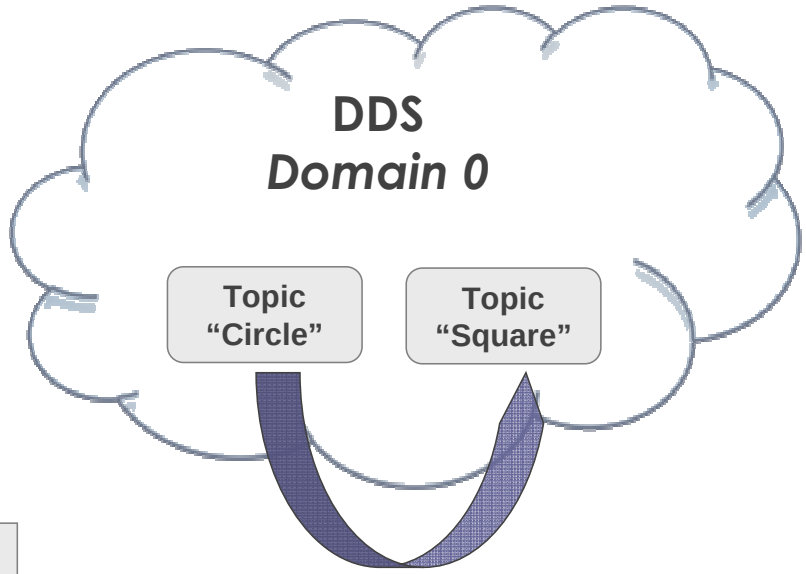
```
new RouteBuilder() {
    @override
    public void configure() {
        from("ddsi:Foo:0/MyType").to("ddsi:Bar:1/MyType");
    }
}
```



Copyright PrismTech 2012

DDS Topic Bridging

- To send data to another Topic
  - Require same data type



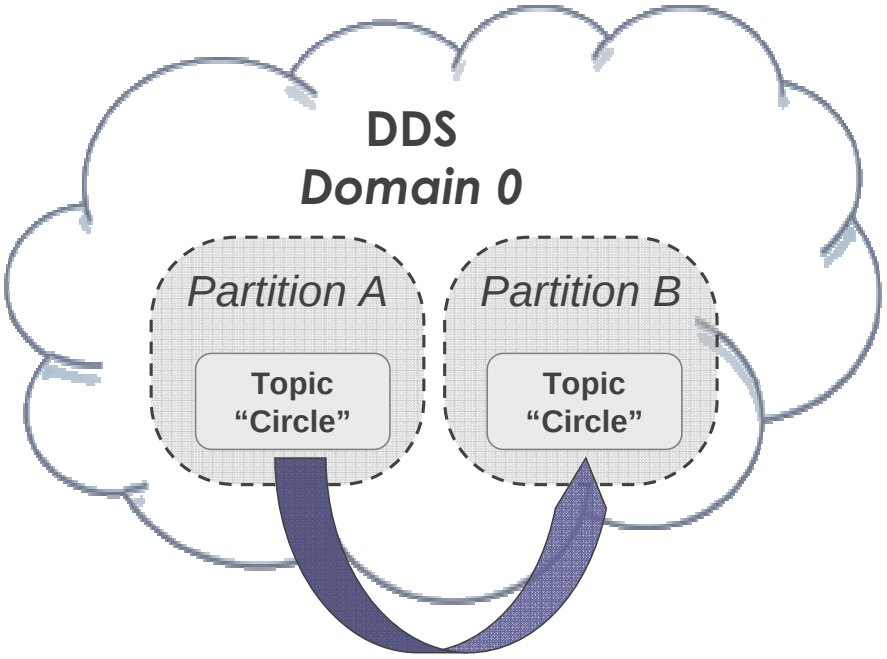

```
from("ddsi:Circle:0/ShapeType")
.to("ddsi:Square:0/ShapeType");
```

OpenSplice DDS

Copyright PrismTech 2012

DDS Partition Bridging

- To send data to another Partition
  - Require same data type




```
from("ddsi:Circle:0/ShapeType?partition=A")
.to("ddsi:Circle:0/ShapeType?partition=B");
```

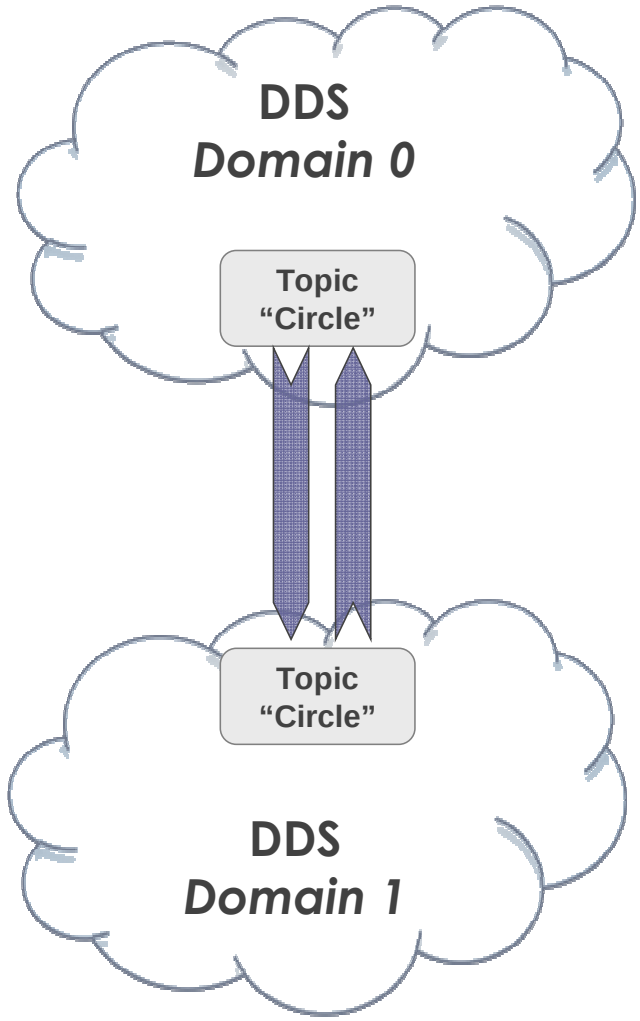
Copyright PrismTech 2012

DDS Domain Bridging

- To integrate 2 (or more) DDS Domains
  - Per Topic bridging
  - Unidirectional or bidirectional

```
from("ddsi:Circle:0/ShapeType")
.to("ddsi:Circle:1/ShapeType");

from("ddsi:Circle:1/ShapeType")
.to("ddsi:Circle:0/ShapeType");
```



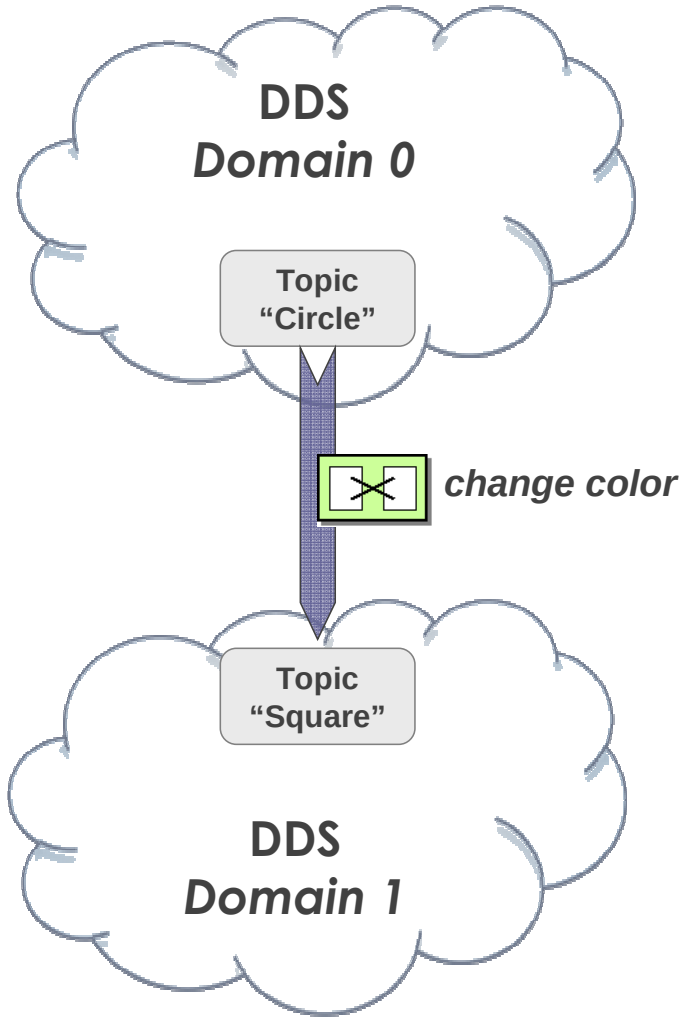
OpenSplice | DDS

Copyright PrismTech 2012

DDS Data transformation

- To change data value
- To transform to another type

```
from("ddsi:Circle:0/ShapeType")
.unmarshal("cdr")
.process(new Processor() {
   public void process(Exchange e) {
      ShapeType shape =
        e.getIn().getBody(ShapeType.class);
      shape.color = "GREEN";
   }
})
.to("ddsi:Square:1/ShapeType");
```

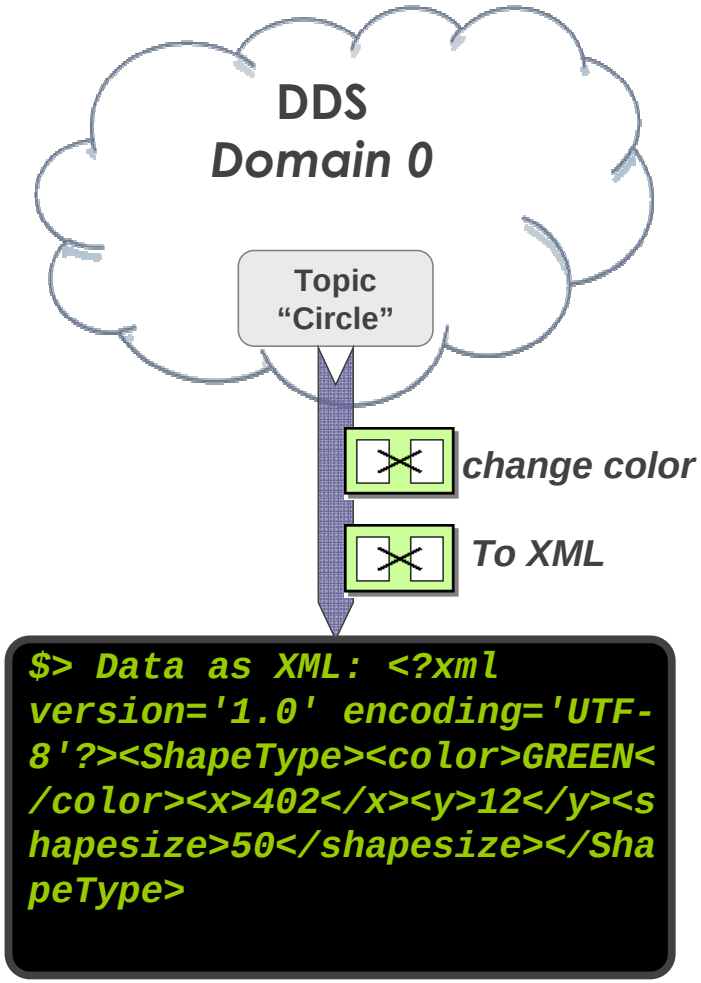


Copyright PrismTech 2012

DDS Data transformation

- Using scripting languages
  - Groovy, Python, JavaScript, XPath, XQuery …
- Or using marshallers/unmarshallers
  - Java Serialization, XML, JSON, SOAP, Protobuf…

```
from("ddsi:Circle:0/ShapeType")
.unmarshal("cdr")
.transform()
  .groovy("request.body.color='GREEN';"+
          "request.body")
.marshal().xstream()
.log("Data as XML: ${body}");
```

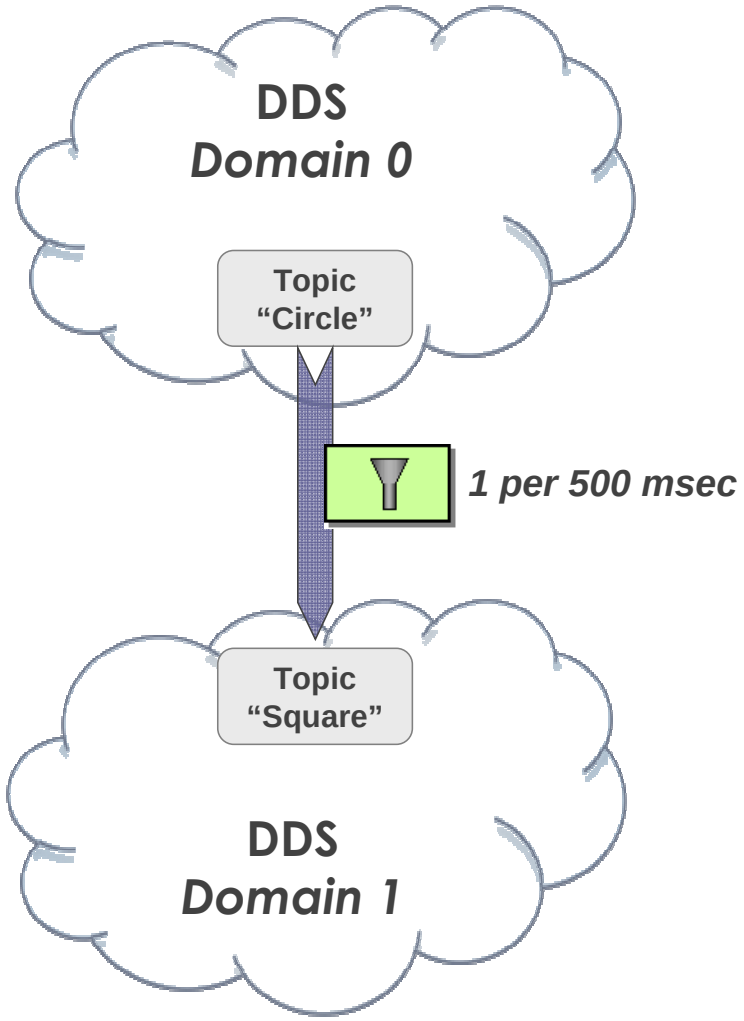


Copyright PrismTech 2012

DDS Data sampling

- To lower publication rate

```
from("ddsi:Circle:0/ShapeType")
.sample(500, TimeUnit.MILLISECONDS)
.to("ddsi:Square:1/ShapeType");
```

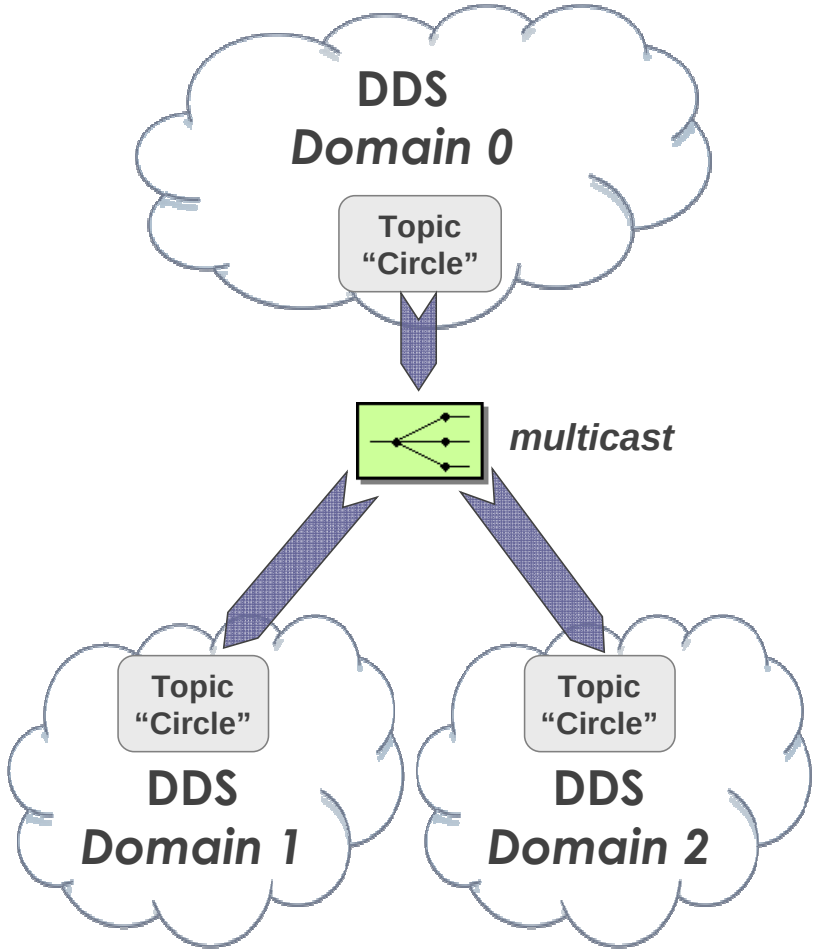


Copyright PrismTech 2012

DDS Domains multicast

- To send DDS data to multiple Domains (or Topics)

```
from("ddsi:Circle:0/ShapeType")
.multicast().parallelProcessing()
.to("ddsi:Circle:1/ShapeType",
    "ddsi:Circle:2/ShapeType");
```

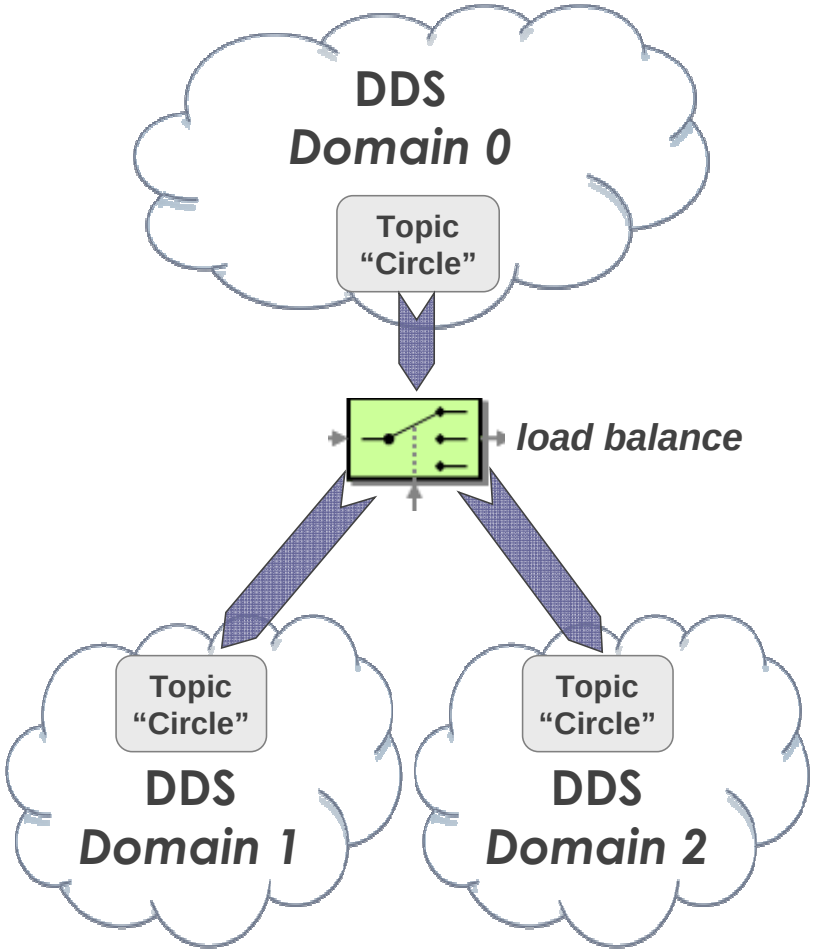


Copyright PrismTech 2012

DDS Domains Load Balancing

- To load balance DDS data to multiple Domains (or Topics)

```
from("ddsi:Circle:0/ShapeType")
.loadBalance().roundRobin()
.to("ddsi:Circle:1/ShapeType",
    "ddsi:Circle:2/ShapeType");
```

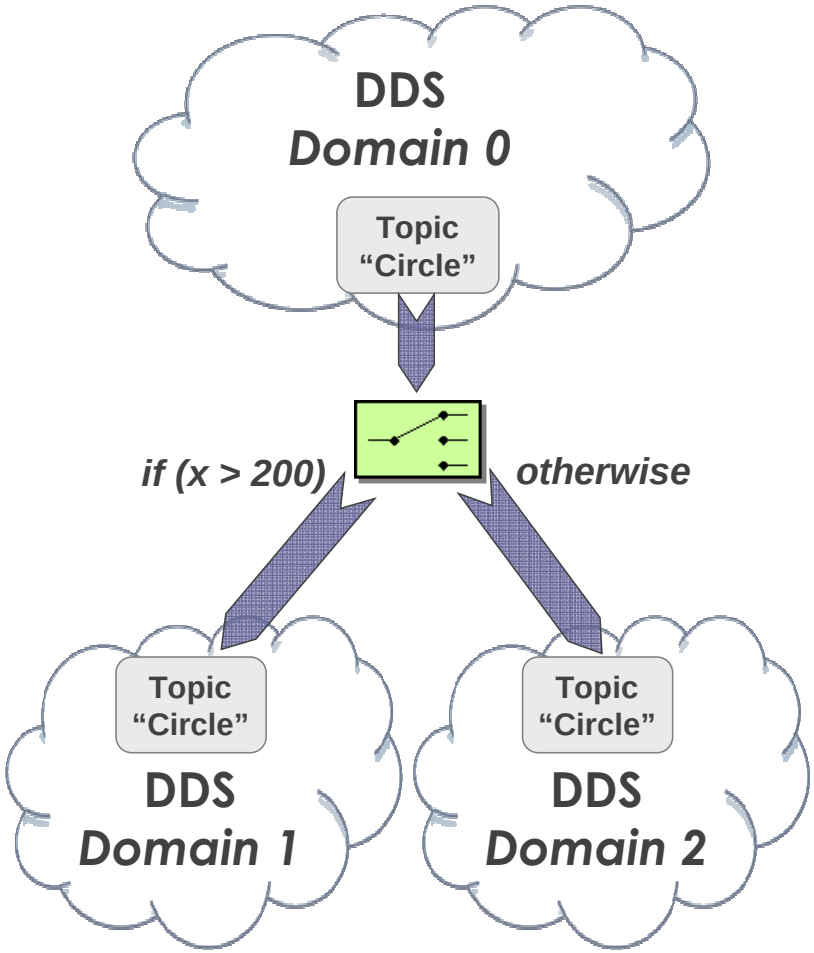


Copyright PrismTech 2012

Content-based routing

- To route DDS data depending its value

```
from("ddsi:Circle:0/ShapeType")
.unmarshal("cdr")
.choice()
   .when()
   .groovy("request.body.x>200")
      .to("ddsi:Circle:1/ShapeType")
   .otherwise()
      .to("ddsi:Circle:2/ShapeType");
```

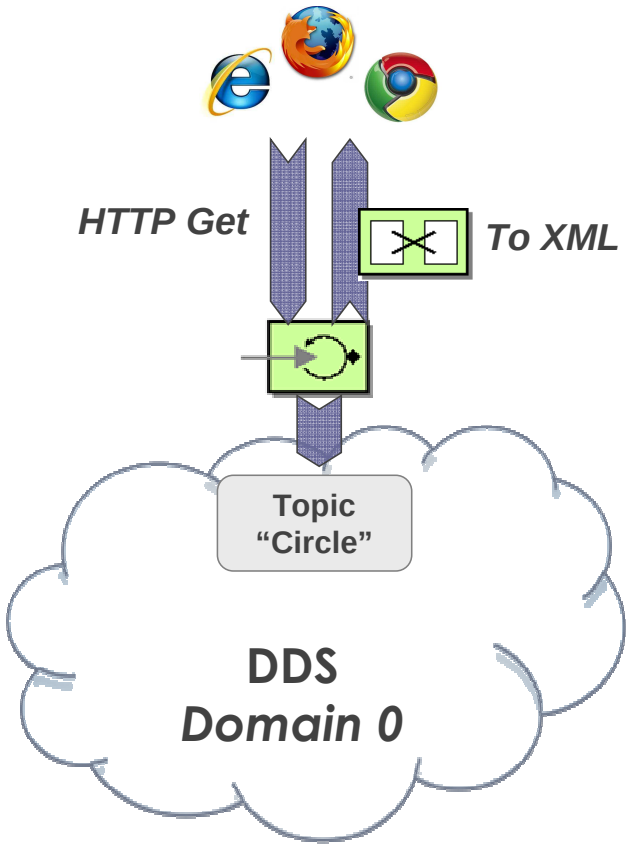


Copyright PrismTech 2012

HTTP REST interoperability

- To poll DDS data from HTTP
- To publish DDS data via HTTP

```
from("restlet:http://localhost:4444/circle")
.pollEnrich("ddsi:Circle:0/ShapeType")
.unmarshal("cdr")
.marshal().xstream();
```



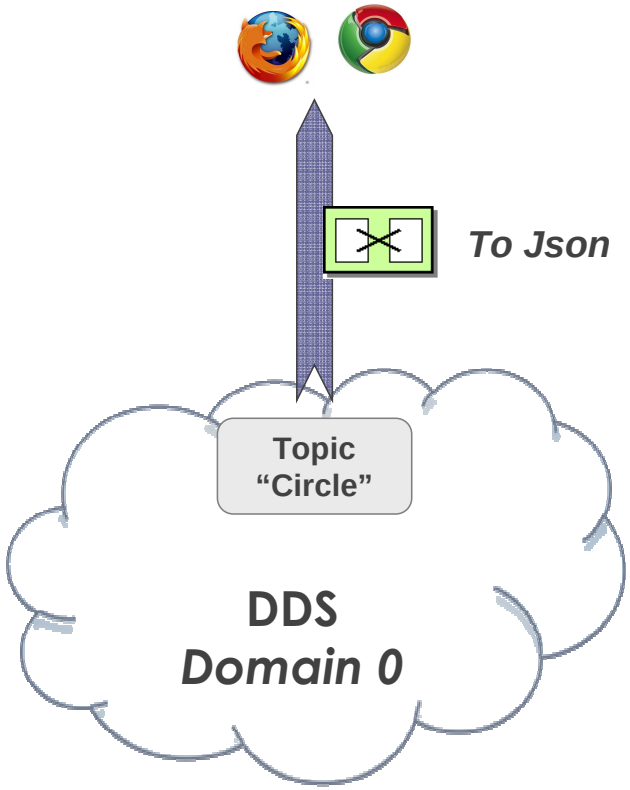
OpenSplice DDS

Copyright PrismTech 2012

WebSocket interoperability

- To push DDS data to a Web browser
  - Requires Camel 2.10

```
from("ddsi:Circle:0/ShapeType")
.unmarshal("cdr")
.marshal().json()
.to("websocket://circle?sendToAll=true");
```

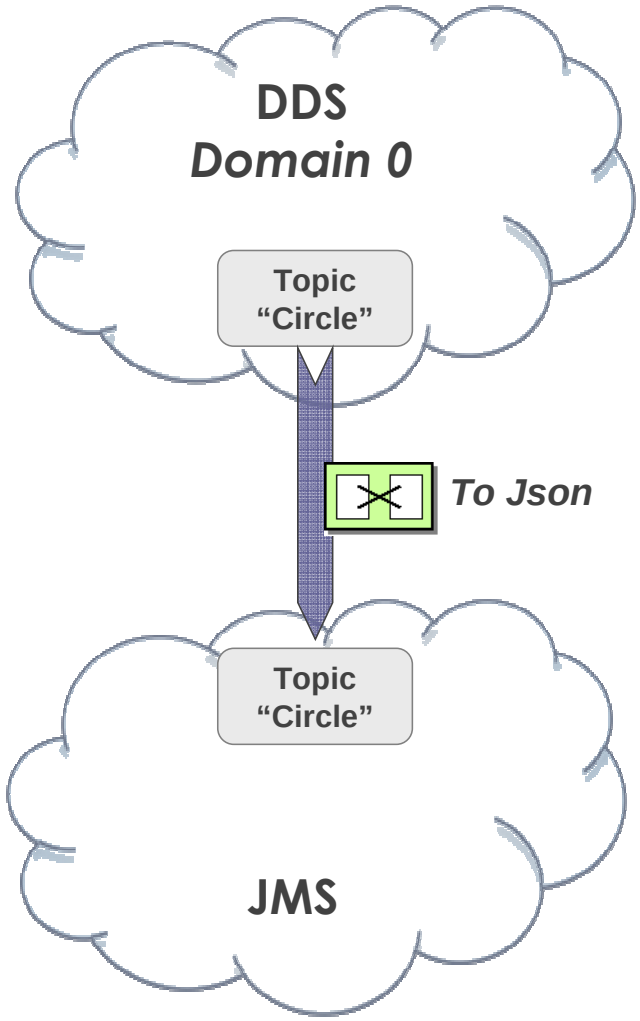


Copyright PrismTech 2012

JMS interoperability

- To send/receive data from/to JMS

```
from("ddsi:Circle:0/ShapeType")
.unmarshal("cdr")
.marshal().json()
.to("jms:topic:circle?jmsMessageType
=Text&deliveryPersistent=false");
```



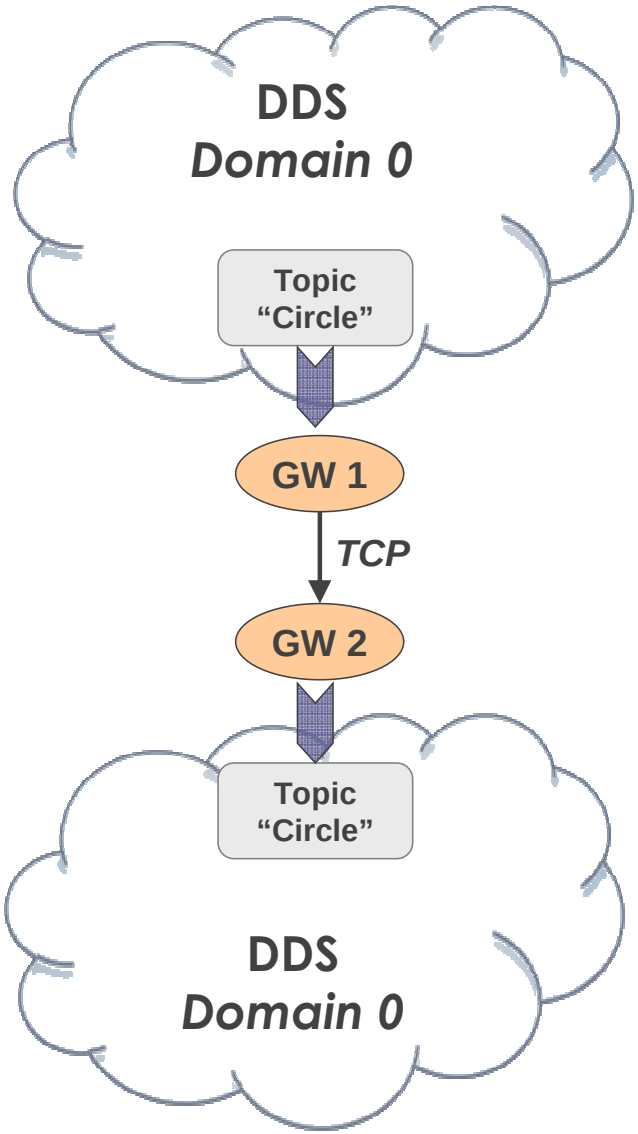
Copyright PrismTech 2012

DDS Domain Bridging – TCP tunnel

- To integrate 2 DDS Domains, via TCP (or UDP) tunnel
  - Per Topic bridging
  - Unidirectional or bidirectional
  - Possibly adding SSL

```
// on GW1:
from("ddsi:Circle:0/ShapeType")
.to("netty:tcp://localhost:6789?sync=false");
```

```
// on GW2:
from("netty:tcp://localhost:6789?sync=false")
.to("ddsi:Circle:0/ShapeType");
```



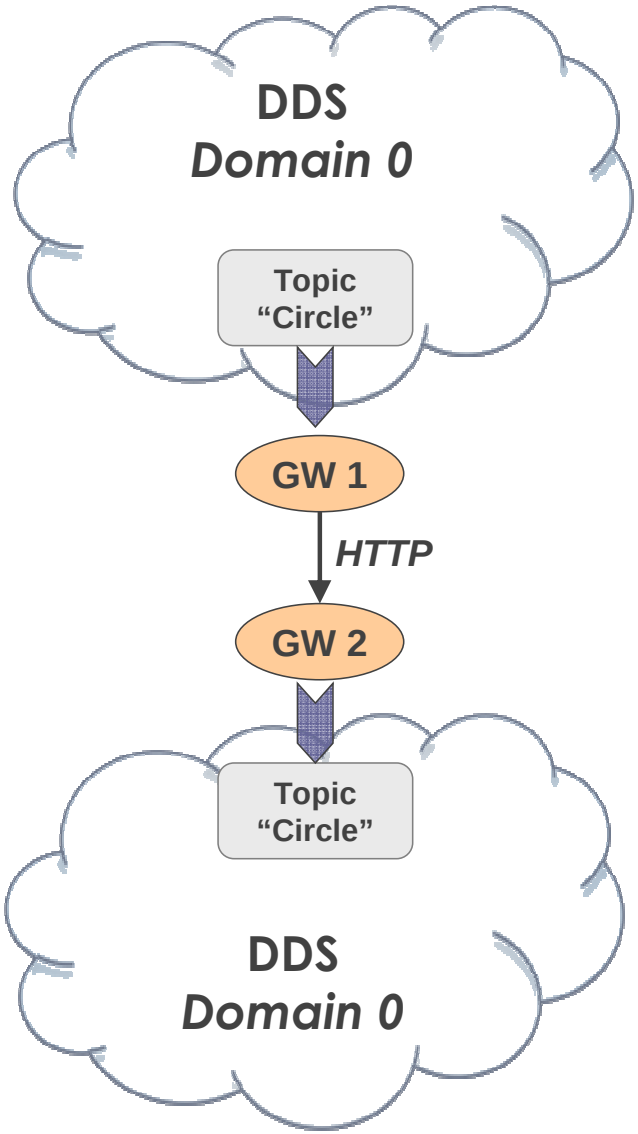
Copyright PrismTech 2012

DDS Domain Bridging – HTTP tunnel

- To integrate 2 DDS Domains, via HTTP(or HTTPS) tunnel
  - Per Topic bridging
  - Unidirectional or bidirectional

```
// on GW1:
from("ddsi:Circle:0/ShapeType")
.unmarshal("cdr")
.marshal().json()
.to("jetty:http://localhost:5001/circle");
```

```
// on GW2:
from("jetty:http://localhost:5001/circle")
.setExchangePattern(ExchangePattern.InOnly)
.unmarshal().json()
.marshal("cdr")
.to("ddsi:Circle:0/ShapeType");
```



Copyright PrismTech 2012

Concluding remarks

- OpenSplice Gateway allows easy integration of DDS-based systems and/or subsystems by:
  - routing DDS data between Topics, Partitions and Domains
  - transforming data on the fly
  - changing data format on the fly (XML, Json…)
  - sending/receiving data to/from various technologies
  - allowing tunnelling via TCP, HTTP, HTTPS…

Copyright PrismTech 2012