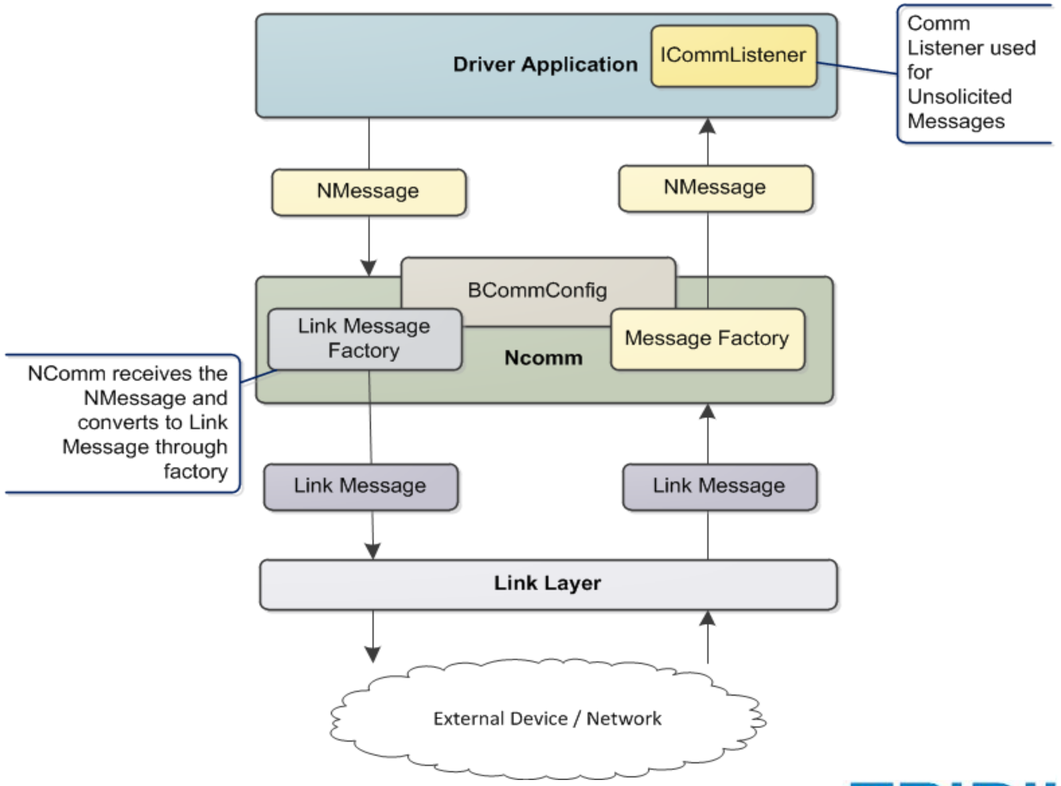
NF 23

CONNECTING THE WORLD

NDriver – with a library

Jason Woollard / Nick Dodd

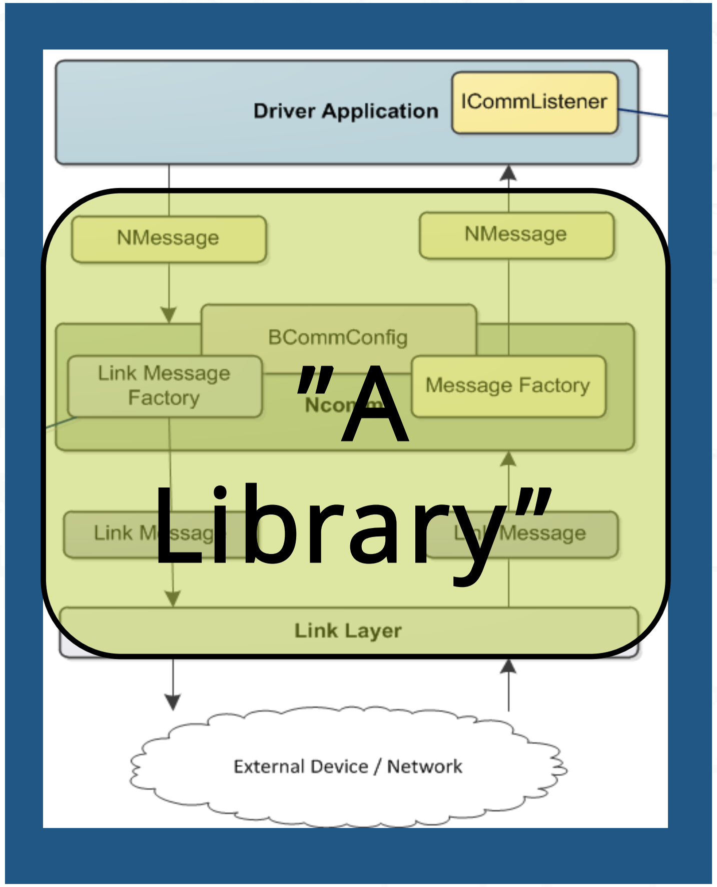


Comm Listener used for Unsolicited Messages

NComm receives the NMessage and converts to Link Message through factory

TRIDIUM

NDriver – with a library?



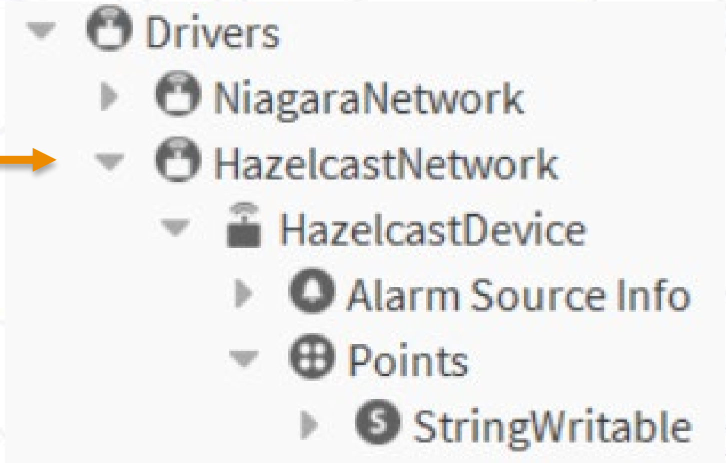
- Typically, the New Driver wizard anticipates developers will be parsing a protocol into Niagara's Object Model

- Anticipates modelling via LinkMessage and NMessage implementations

- What if you 'simply' wish to use an off-the-shelf library?

You're assuming I know what NDriver is?

- Niagara's current Driver Framework
- Includes features for modelling diverse protocols e.g.
  - Unsolicited Messages,
  - Retries, etc.
- Common look and feel
- Models points / schedules / alarms / histories



- Drivers
  - NiagaraNetwork
  - HazelcastNetwork
    - HazelcastDevice
      - Alarm Source Info
      - Points
        - StringWritable

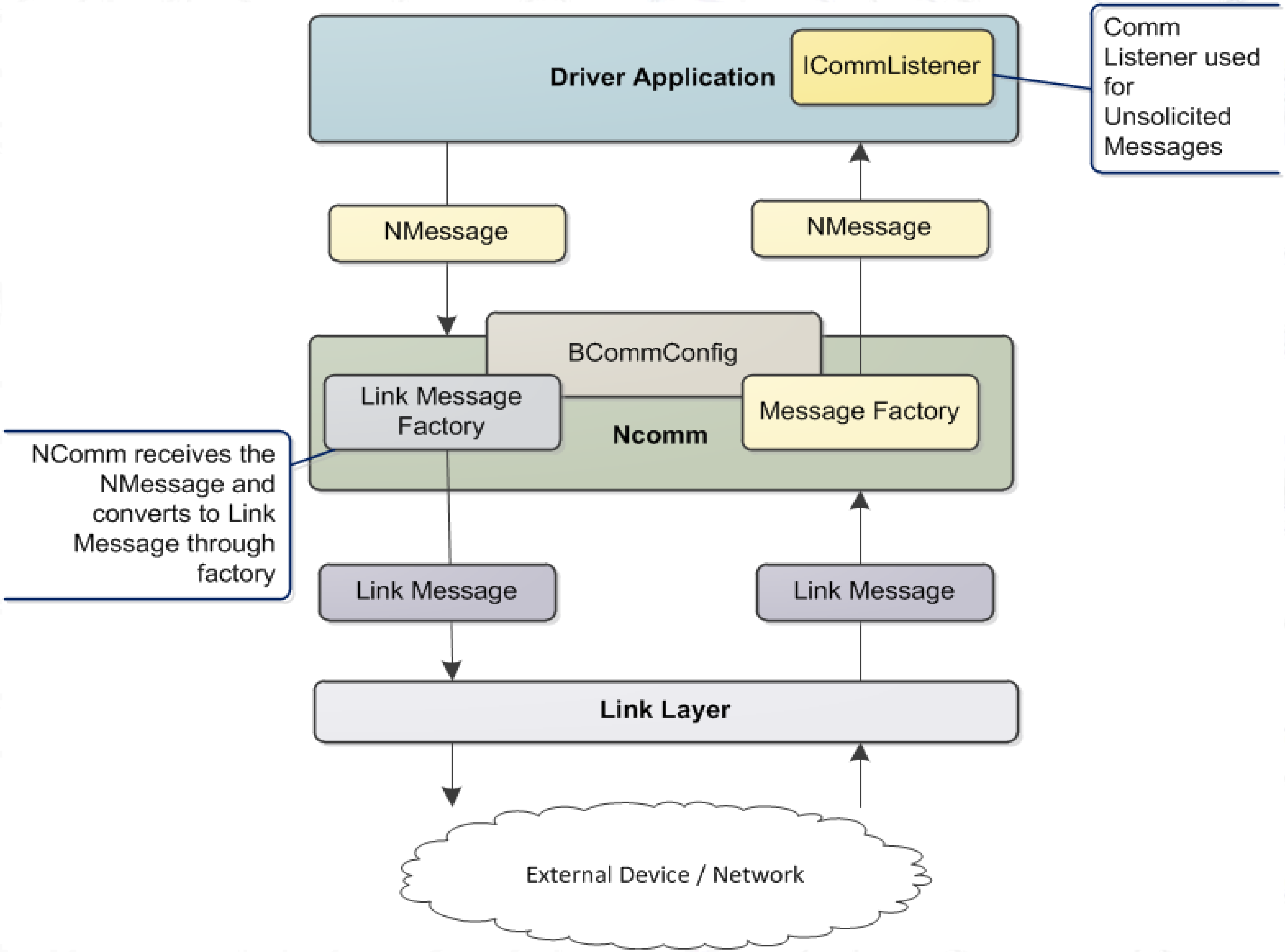
What has NDriver ever done for us?

- Wizard → Template Code
- Discovery
- Auto Manager (wb **and ux** views)
- Worker, Logging, Parsing Utils
- Access to Serial, UDP, TCP, HTTP Link Layers

- X Message: Transactions, Fragmentation, Retries, Timeouts

- Licensing; Training; Support; Marketplace; Community; Security … plus, the rest of the Framework!

The Regular Message…



Comm Listener used for Unsolicited Messages

NComm receives the NMessage and converts to Link Message through factory

Message in a Model

- **NMessage:** Read / Write / Connect / Ping / Heartbeat / Responses
  - Mapping to application logic
  - Many classes

```
public class FooMessage extends NMessage {
  int   fooMsgType = MY_TYPE;
  int   tag;
  int   prop1;
  float prop2;

  @Override
  public void fromInputStream (InputStream is) {}
  @Override
  public boolean toOutputStream  (OutputStream os) {
return true; }
```

- **LinkMessage:**
  - byte[] wrapped for transport
  - Single class

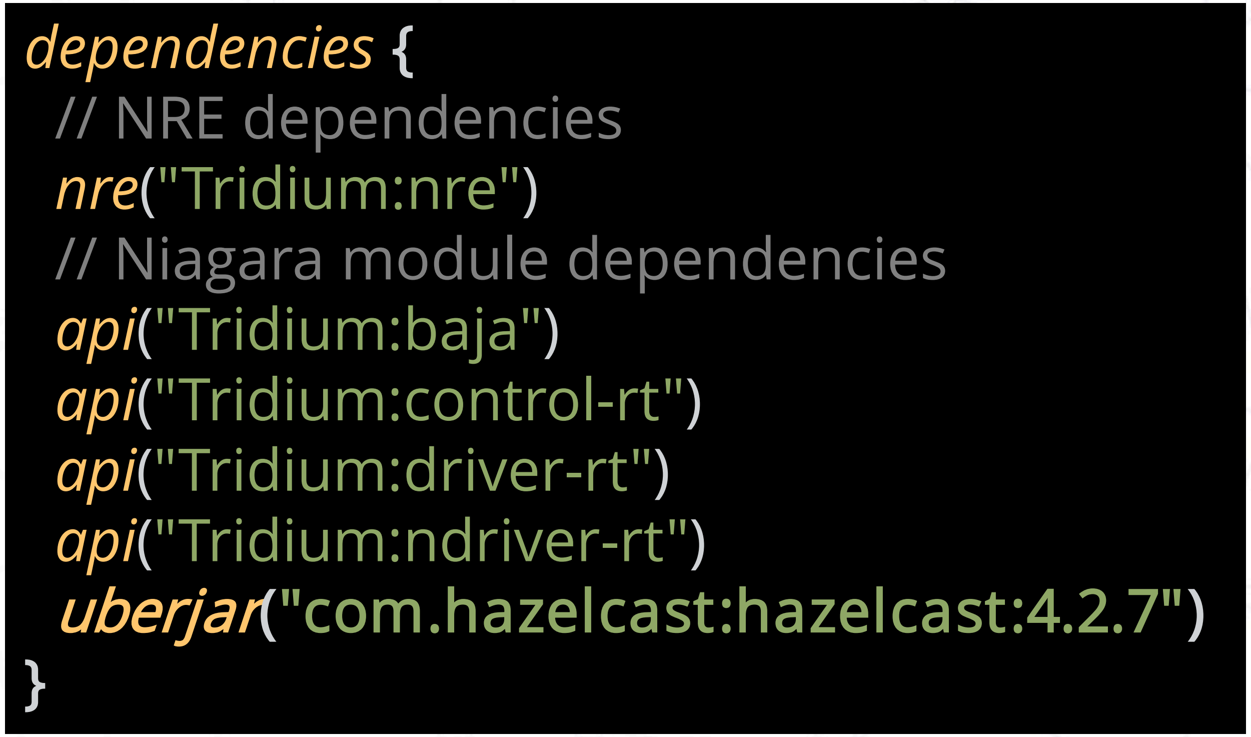
```
public class FooSerialLinkMessage  extends LinkMessage {
  byte[] data;

  @Override
  public void setMessage(NMessage msg) {}

  @Override
  public boolean receive (InputStream is) { return false; }
```

- |0, 70, 80, **23** |0, 34, 54, **23** | 0, 2…

TRIDIUM

I can use libraries?

- Sure, **we** do it all the time (see **lib/readmeLicenses.txt**)
- Uberjar via myModule-rt.gradle.kts e.g.



```
dependencies {
  // NRE dependencies
  nre("Tridium:nre")
  // Niagara module dependencies
  api("Tridium:baja")
  api("Tridium:control-rt")
  api("Tridium:driver-rt")
  api("Tridium:ndriver-rt")
  uberjar("com.hazelcast:hazelcast:4.2.7")
}
```

// Supported dependency types:
module://docDeveloper/doc/build.html#dependencies

Are you _*sure*_ I can use libraries?

- Security Says, No?
- As mentioned in the "news" we have ever increasing permissions available for developers compared to 4.0
- In the case of hazlecast-rt it required:

GET_ENVIRONMENT_VARIABLES

MANAGE_EXECUTION

REFLECTION

MBEAN_PERMISSION

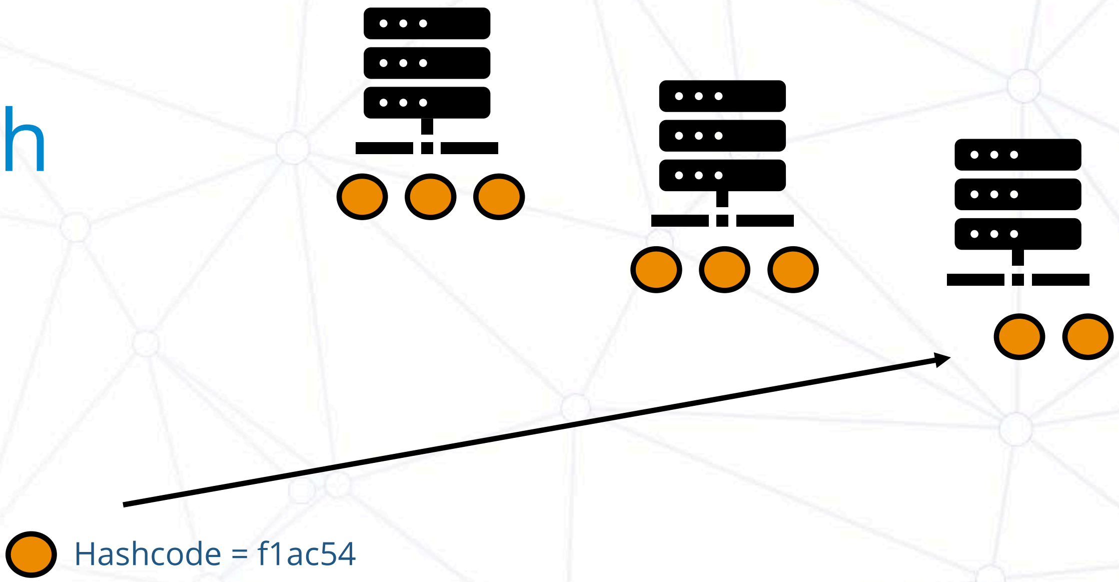
NETWORK_COMMUNICATION

Tell me about this *library* !

- Sorry – the least important part of the presentation ☺
- Could have been anything:
  - Procrastination scheduler,
  - NTP
  - Our favourite… another Traffic Light!
- We chose Hazlecast
  - ‘A Distributed in-memory data grid’
  - aka: a rather funky HashMap

Hazelwhat?

- Clustered Maps
    - *IMap<K,V>* - any Java Object
- Values distributed in cluster
    - Cluster rebalance – fault tolerance
- Read through / write through
    - Front a real data source with *MapStore*
- Docker - scaling



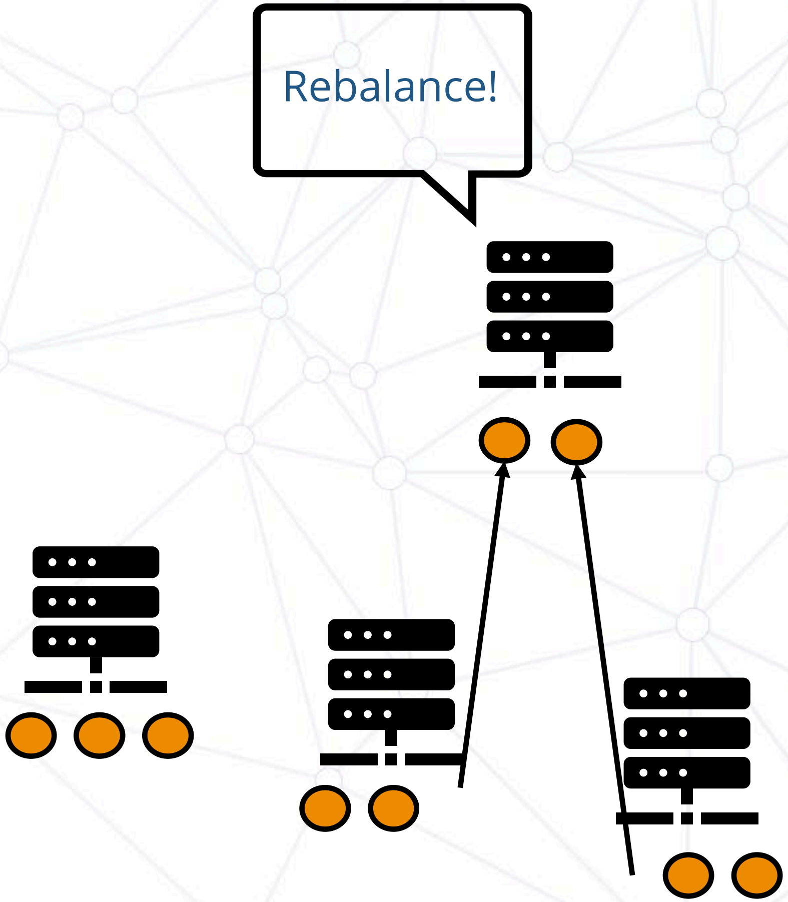
Hazelwhat?

- Clustered Maps
  - *IMap<K,V>* - any Java Object

- Values distributed in cluster
  - Cluster rebalance – fault tolerance

- Read through / write through
  - Front a real data source with *MapStore*

- Docker - scaling

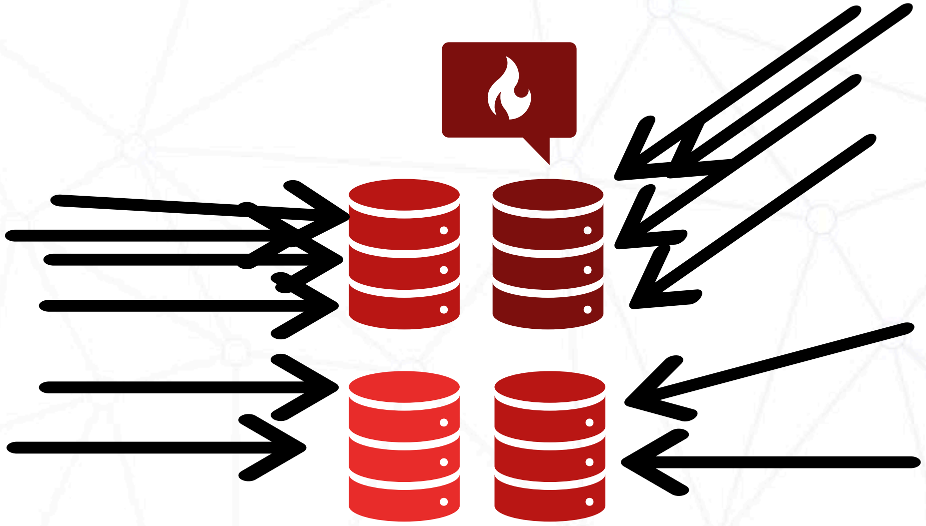


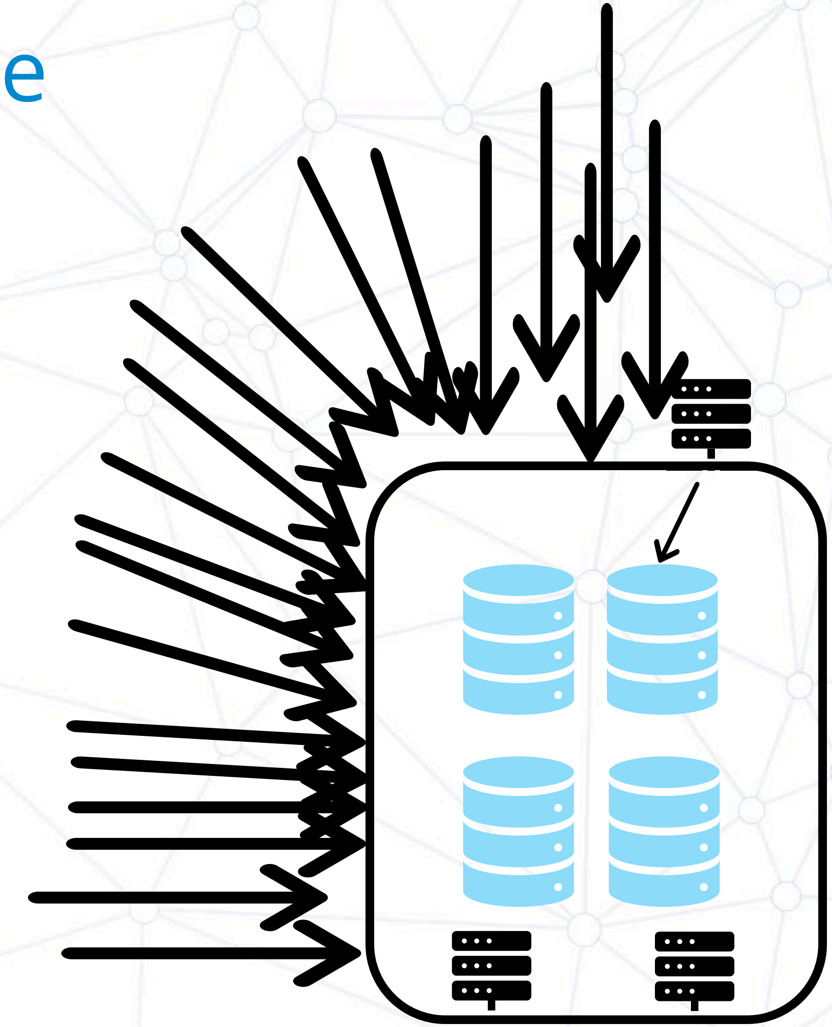
NF 23

TRIDIUM

Hazelwhy?

- Cache - Keep the heat off your database
- Session data
- Key-value database
- Data processing





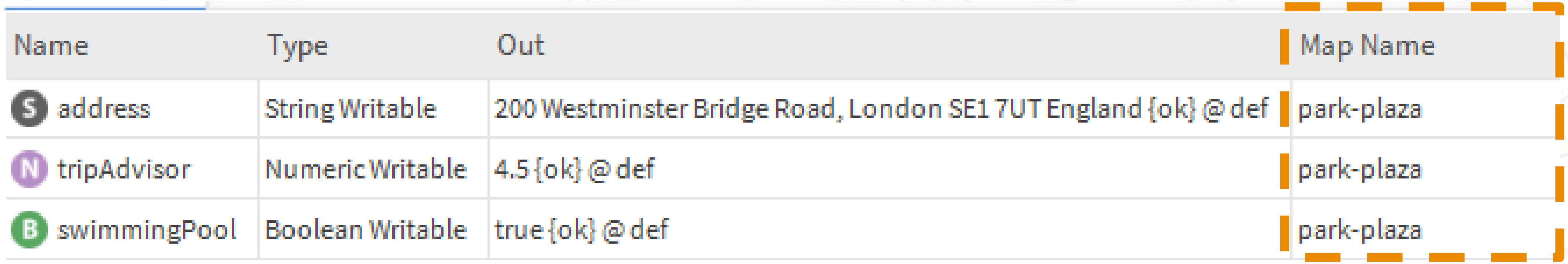
Hazelhow?

- HazelcastInstance
  - Join cluster
  - Get distributed objects
  - Configuration
- IMap
  - A HashMap with teeth

```
HazelcastInstance hz = HazelcastClient.newHazelcastClient();

IMap map = hz.getMap("my-distributed-map");

// regular HashMap stuff
map.put("key", "value");
map.get("key");
map.putIfAbsent("somekey", "somevalue");
map.replace("key", "value", "newvalue");

// eyes emoji
map.lock("key", 5, TimeUnit.SECONDS);
map.evict(“otherKey");
map.putAsync("key", "value", 5, TimeUnit.SECONDS);
```

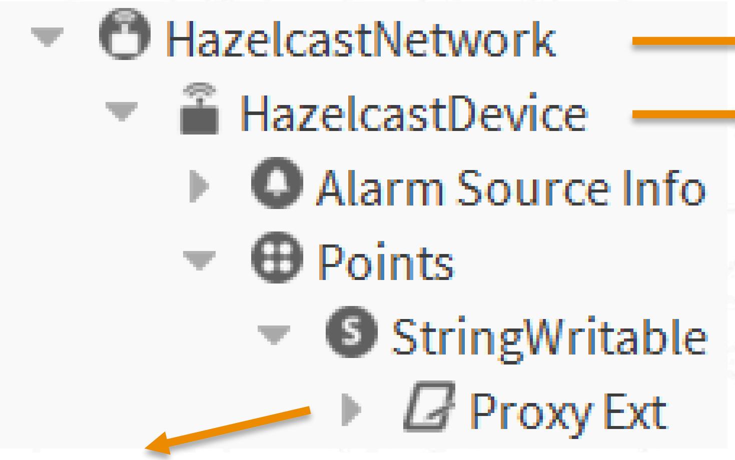
Explain the basic model, please!

- Device - Hazelcast ‘Cluster’
- Map Points to a Single value from Hazlecast
  - Uses Point Name for key (*not suggested/conventional!*)
- ProxyExt - Specifies which Map to look in
  - Read + Write supported

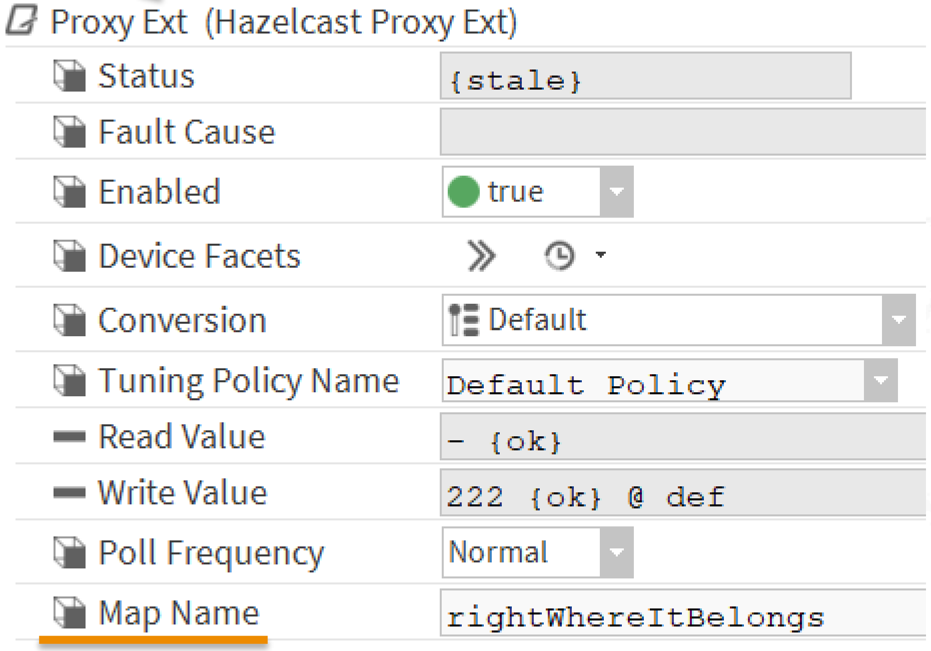


Name	Type	Out	Map Name
S address	String Writable	200 Westminster Bridge Road, London SE1 7UT England {ok} @ def	park-plaza
N tripAdvisor	Numeric Writable	4.5 {ok} @ def	park-plaza
B swimmingPool	Boolean Writable	true {ok} @ def	park-plaza

Another Overview

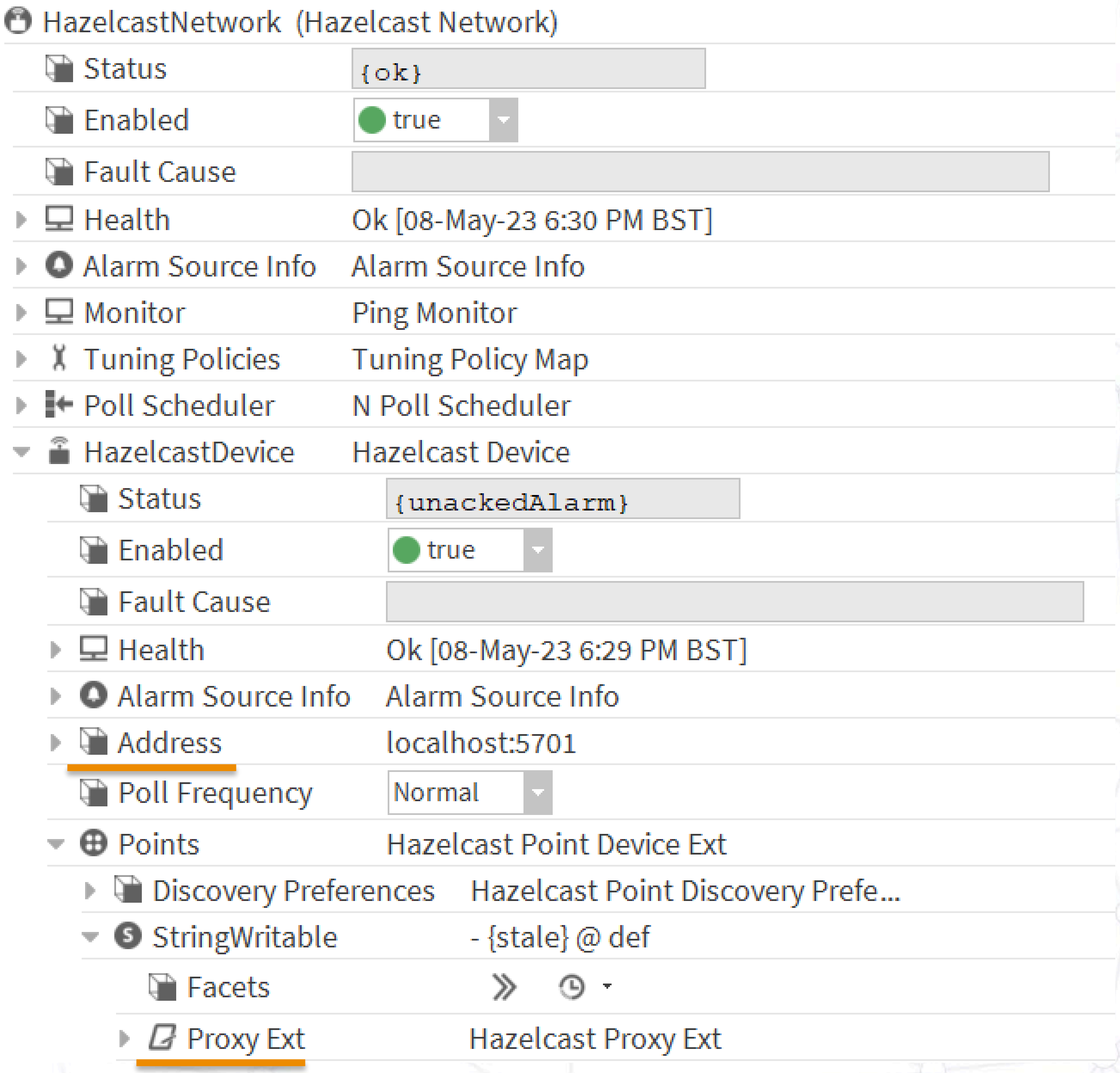


- HazelcastNetwork
  - HazelcastDevice
    - Alarm Source Info
    - Points
      - StringWritable
        - Proxy Ext



Proxy Ext (Hazelcast Proxy Ext)

Status	{stale}
Fault Cause	
Enabled	● true
Device Facets	» ◷ ▾
Conversion	Default
Tuning Policy Name	Default Policy
Read Value	- {ok}
Write Value	222 {ok} @ def
Poll Frequency	Normal
Map Name	rightWhereItBelongs

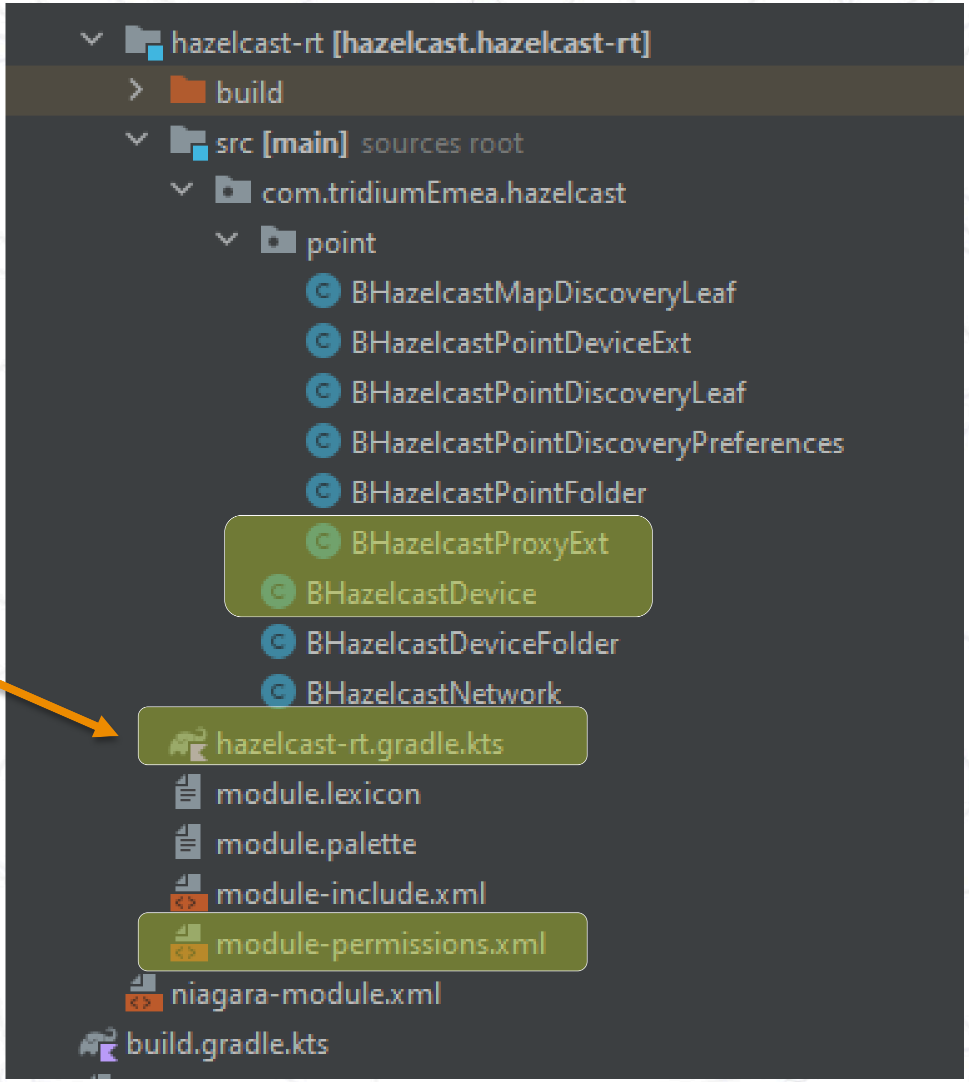


HazelcastNetwork (Hazelcast Network)

Status	{ok}
Enabled	● true
Fault Cause	
Health	Ok [08-May-23 6:30 PM BST]
Alarm Source Info	Alarm Source Info
Monitor	Ping Monitor
Tuning Policies	Tuning Policy Map
Poll Scheduler	N Poll Scheduler
HazelcastDevice	Hazelcast Device
Status	{unackedAlarm}
Enabled	● true
Fault Cause	
Health	Ok [08-May-23 6:29 PM BST]
Alarm Source Info	Alarm Source Info
Address	localhost:5701
Poll Frequency	Normal
Points	Hazelcast Point Device Ext
Discovery Preferences	Hazelcast Point Discovery Prefe...
StringWritable	- {stale} @ def
Facets	» ◷ ▾
Proxy Ext	Hazelcast Proxy Ext

Go on - how difficult was it then?

- Run New Driver Wizard
- Deleted:
  - HazlecastMessage.java
  - HazlecastMessageFactory.java
- Uberjar the library via Gradle
- Keep
  - module.palette
  - module-permissions.xml



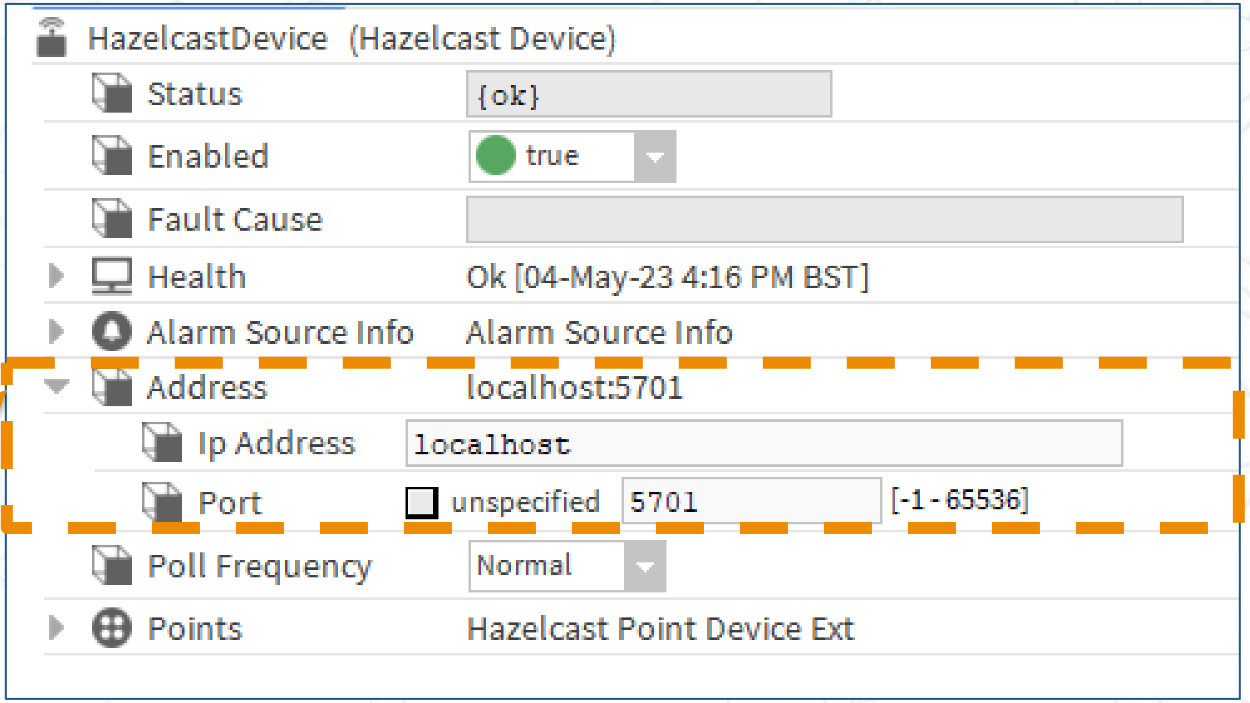
```
hazelcast-rt [hazelcast.hazelcast-rt]
  build
  src [main] sources root
    com.tridiumEmea.hazelcast
      point
        BHazelcastMapDiscoveryLeaf
        BHazelcastPointDeviceExt
        BHazelcastPointDiscoveryLeaf
        BHazelcastPointDiscoveryPreferences
        BHazelcastPointFolder
        BHazelcastProxyExt
      BHazelcastDevice
      BHazelcastDeviceFolder
      BHazelcastNetwork
    hazelcast-rt.gradle.kts
    module.lexicon
    module.palette
    module-include.xml
    module-permissions.xml
  niagara-module.xml
build.gradle.kts
```

NF 23

TRIDIUM

Hazelcast Device

Add a Property for the Ip Address of the service we will connect to:



HazelcastDevice (Hazelcast Device)

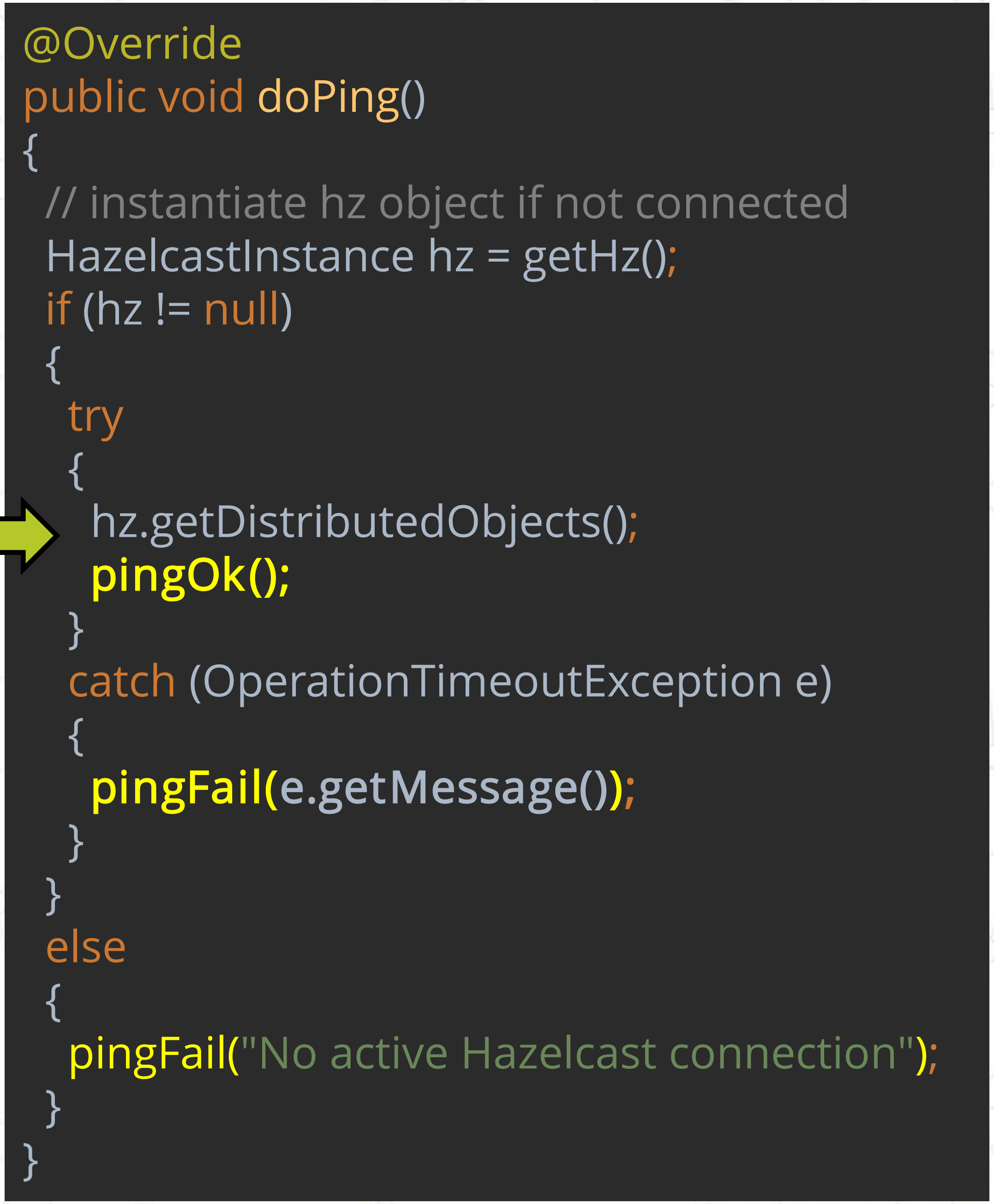
Status	{ok}
Enabled	● true
Fault Cause	
Health	Ok [04-May-23 4:16 PM BST]
Alarm Source Info	Alarm Source Info
Address	localhost:5701
Ip Address	localhost
Port	□ unspecified 5701 [-1 - 65536]
Poll Frequency	Normal
Points	Hazelcast Point Device Ext

```
@NiagaraProperty(
  name = "address",
  type = "BIpAddress",
  defaultValue = "new BIpAddress(\"localhost\",
DEFAULT_HZ_PORT)"
)
```

Hazelcast Device

- doPing is called by the Ping Monitor at a given interval

- getDistributedObjects is a sub-optimal choice from a performance perspective



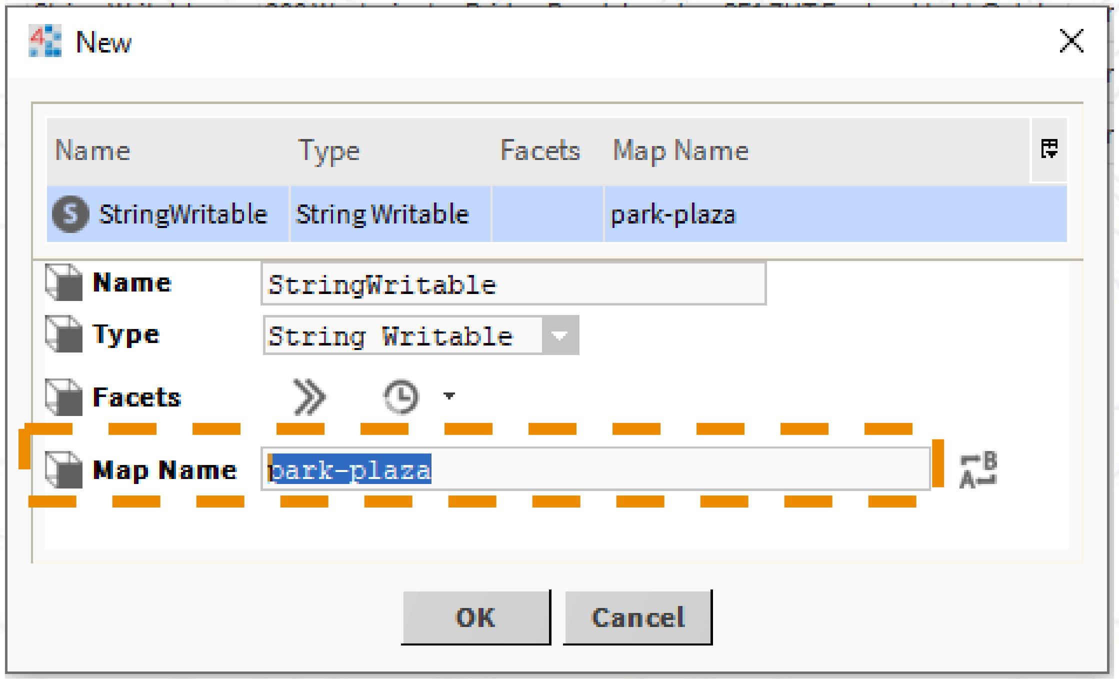
```
@Override
public void doPing()
{
  // instantiate hz object if not connected
  HazelcastInstance hz = getHz();
  if (hz != null)
  {
    try
    {
      hz.getDistributedObjects();
      pingOk();
    }
    catch (OperationTimeoutException e)
    {
      pingFail(e.getMessage());
    }
  }
  else
  {
    pingFail("No active Hazelcast connection");
  }
}
```

Hazelcast ProxyExt

Added properties:

- pollFrequency
- mapName

Manager Facet to make *mapName* editable



Name	Type	Facets	Map Name
StringWritable	String Writable		park-plaza

Name: StringWritable

Type: String Writable

Facets

Map Name: park-plaza

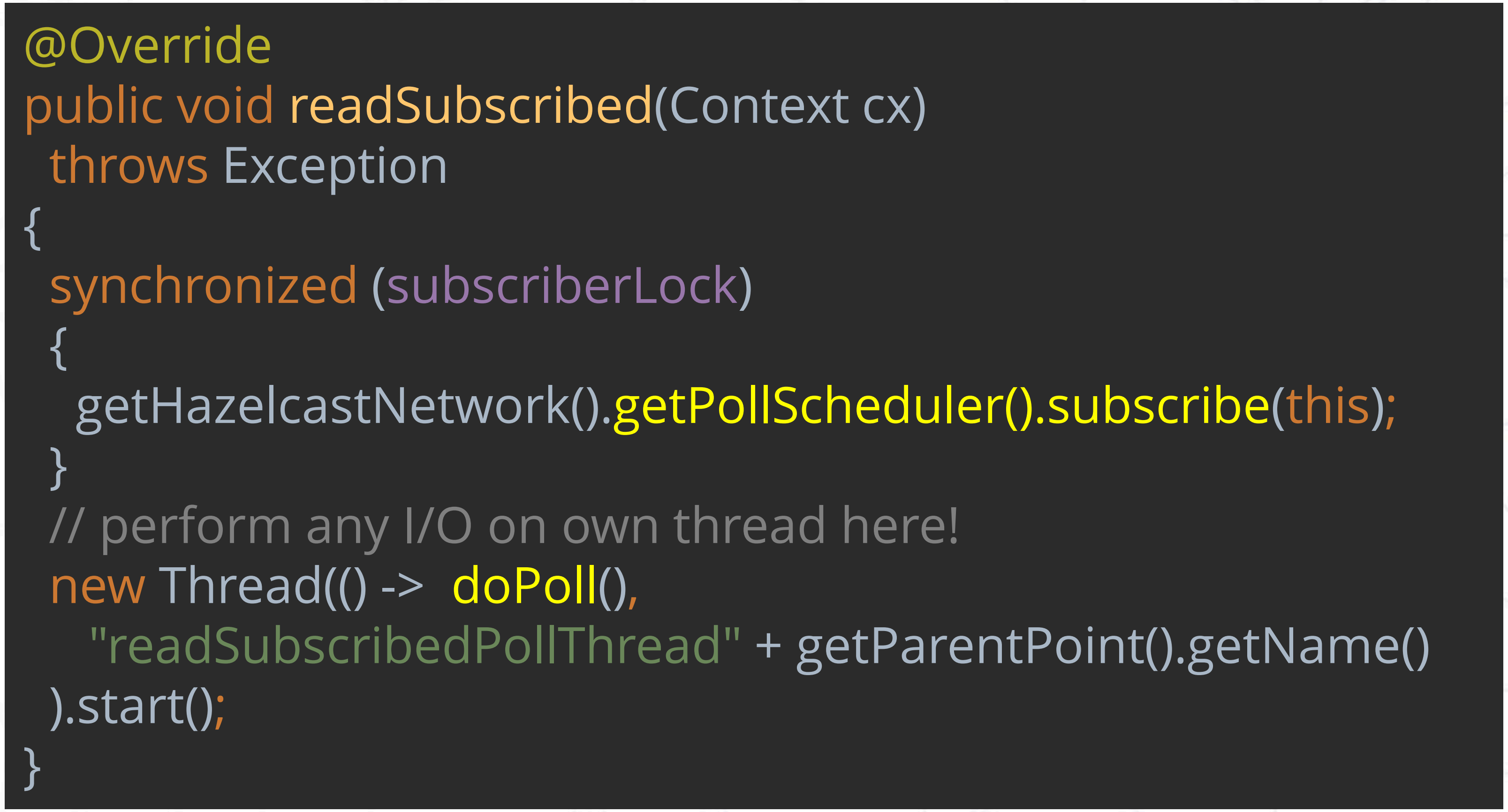
```
@NiagaraProperty(
  name = "mapName",
  type = "String",
  defaultValue = "BString.DEFAULT",
  facets = @Facet("SfUtil.incl(SfUtil.MGR_EDIT)")
)
```

NF 23

TRIDIUM

Keep on polling

Point has been subscribed – register for changes



```
@Override
public void readSubscribed(Context cx)
  throws Exception
{
  synchronized (subscriberLock)
  {
    getHazelcastNetwork().getPollScheduler().subscribe(this);
  }
  // perform any I/O on own thread here!
  new Thread(() ->  doPoll(),
    "readSubscribedPollThread" + getParentPoint().getName()
  ).start();
}
```

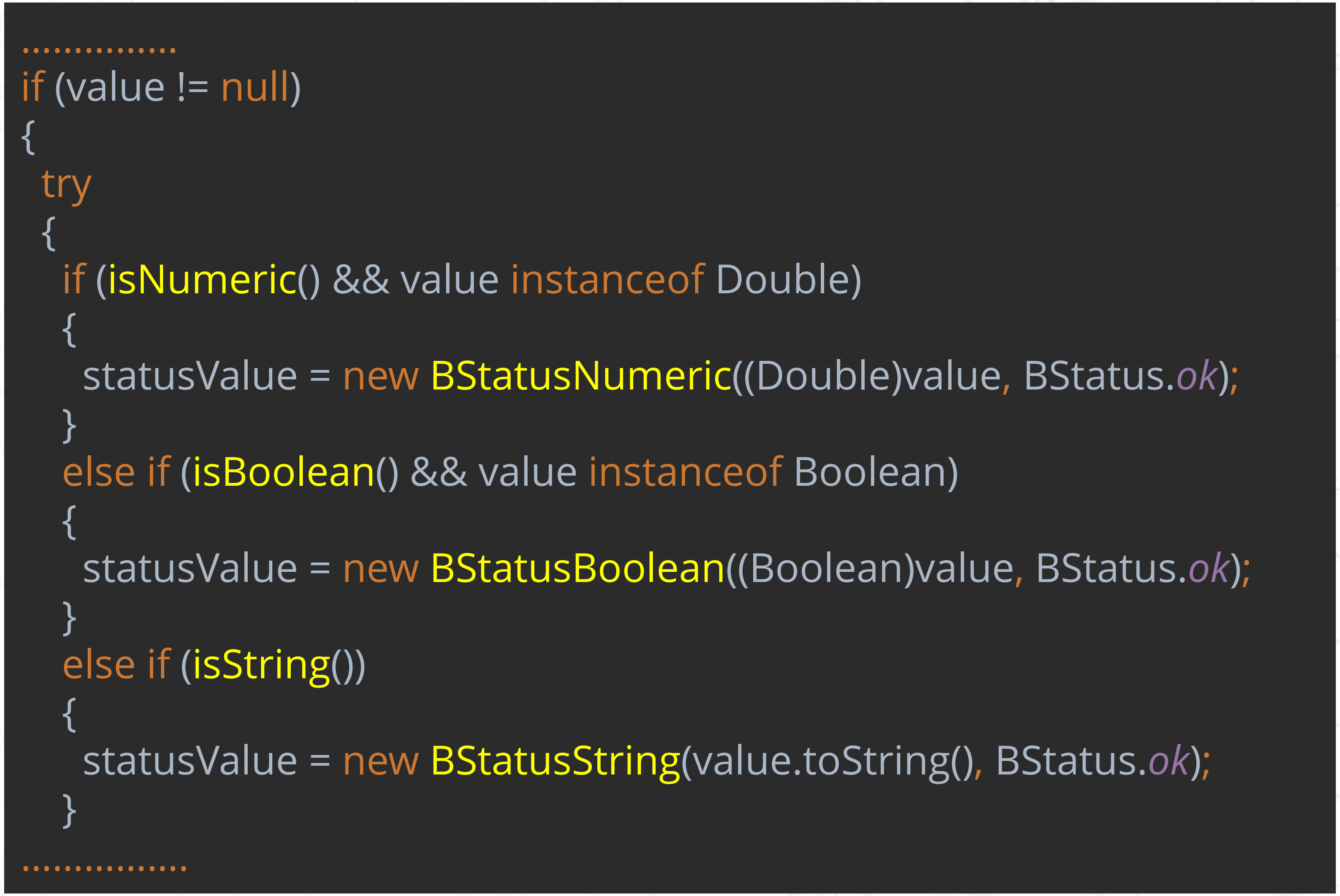
! Remember to unsub in readUnsubscribed() !

Poll – the Hazelcast bit

Get the value out the IMap

```
@Override
public void doPoll()   // runs on poll scheduler thread
{
  HazelcastInstance hz = getBHazelcastDevice().getHz();
  // get the map
  IMap<String, Object> map = hz.getMap(getMapName());
  String key = getParentPoint().getName();
  // get the value  out the map
  Object value = map.get(key);
  ................
```

Poll – convert to the Common Object Model



```
...............
if (value != null)
{
  try
  {
    if (isNumeric() && value instanceof Double)
    {
      statusValue = new BStatusNumeric((Double)value, BStatus.ok);
    }
    else if (isBoolean() && value instanceof Boolean)
    {
      statusValue = new BStatusBoolean((Boolean)value, BStatus.ok);
    }
    else if (isString())
    {
      statusValue = new BStatusString(value.toString(), BStatus.ok);
    }
................
```

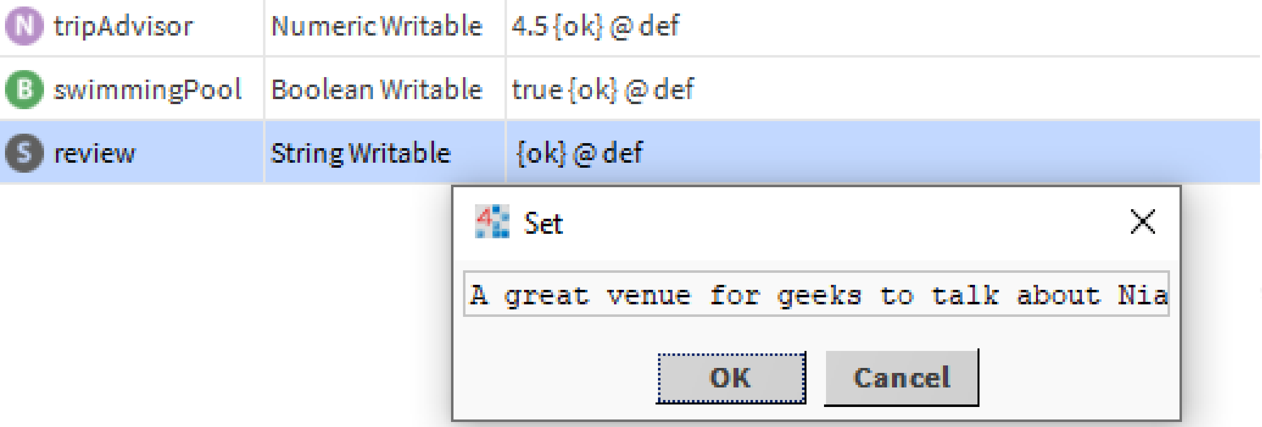
Poll – callbacks

Update the point with the read value

```
................
  if (statusValue != null)
  {
    readOk(statusValue);
  }
  else
  {
    readFail(LEX.getText("hz.point.read.no.value"));
  }
}
catch (Exception e)
{
  readFail(LEX.getText("hz.point.read.failed“, e.getMessage()));
}
```

Write()

Can be invoked by the user, or by other code



N tripAdvisor	NumericWritable	4.5 {ok} @ def
B swimmingPool	BooleanWritable	true {ok} @ def
S review	StringWritable	{ok} @ def

Set

A great venue for geeks to talk about Nia

```
@Override
public BReadWriteMode getMode()
{
  return BReadWriteMode.readWrite;
}
```

Perform I/O on your own thread

```
@Override
public boolean write(Context cx)
  throws Exception
{
  new Thread(() -> {

   ………………………………

  }, "writeSaidThread-" +
getParentPoint().getName()).start();

  return true;
}
```

Return true if a write is now pending

Write

Get the map

```
IMap<String, Object> map =
hz.getMap(getMapName());
String key = getParentPoint().getName();
```

What are we writing?

```
BStatusValue writeValue = getWriteValue();
```

Write value to the IMap

```
if (isBoolean())
{
  map.put(key, writeValue.getValueValue().as(BBoolean.class).getBoolean());
}
```

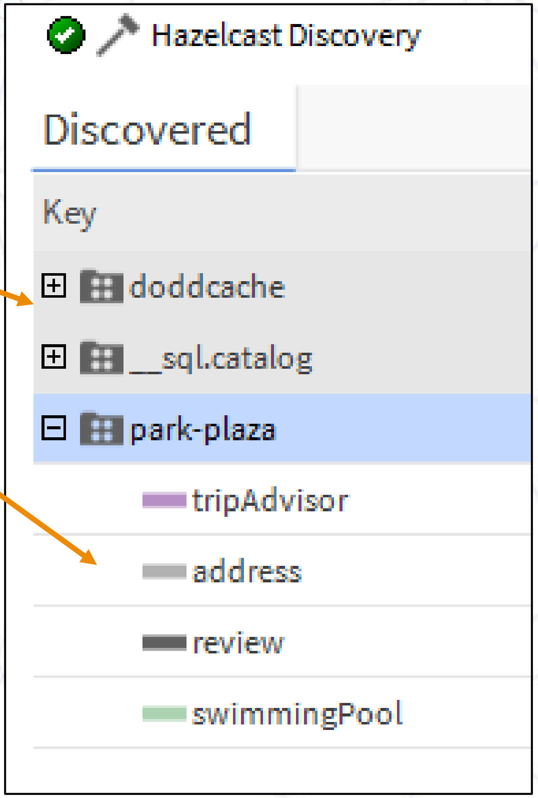
Callback to point

```
  writeOk(writeValue);
}
catch (IOException e)
{
  writeFail(e.getMessage());
}
```

Adding discovery…

Override *getDiscoveryObjects()* in BHazelcastPointDeviceExt

- < 20 lines of code
- *hz.getDistributedObjects()* to get all maps
- Convert each to a *BNDiscoveryGroup (folder)*
- Add each map value as child *BNPointDiscoveryLeaf*



Hazelcast Discovery

Discovered

Key

- doddcache
- __sql.catalog
- park-plaza
  - tripAdvisor
  - address
  - review
  - swimmingPool

No need to override manager views, no wb or ux module!!

Summary

- **Device**: doPing(); add BIpAddress property
- **Proxy Ext**: doPoll(); write(); add 2 properties for Address and Frequency
- No changes to Network; Device Folder; Point Folder
- Which Library to Uberjar? Please don’t ask us ☺
  - Someone already has 10 business ideas off the research for this!

NF
23

CONNECTING
THE WORLD