Simulation Interoperability Standards Organization

*"Simulation Interoperability & Reuse through Standards"*

# News in HLA 4 Object Modelling for Developers

***2024-SIW-Presentation-18***

***Mikael Karlsson, Pitch Technologies, Sweden***

# HLA Evolved to HLA 4

This presentation provides a detailed look at the changes from HLA Evolved to HLA 4 from a developer perspective.

We look at changes in Object Modelling, including merge rules, and how they can be used to improve FOM modelling.

This presentation does not cover the history of simulation, policy making, etc. This is for federate (and RTI) developers.

# Unsigned Integers

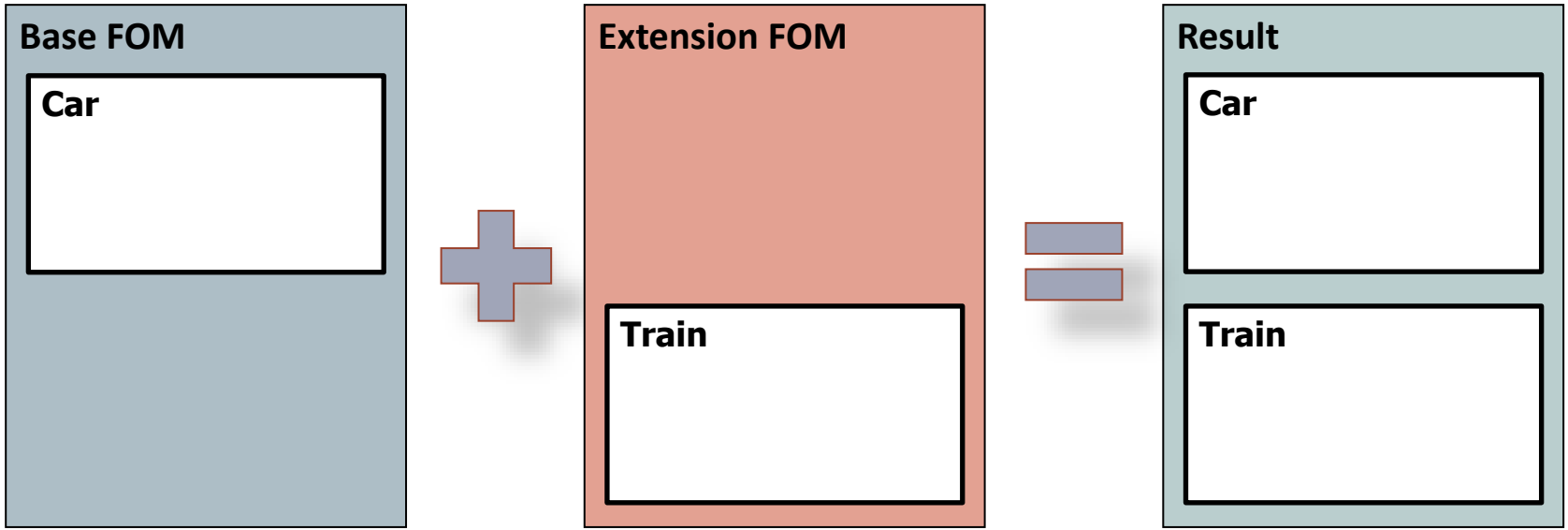
HLA 4

Unsigned data types (that match existing signed types) have been added to FOM and Encoding Helpers.

| In HLA Evolved | Added in HLA 4 |
|---|---|
| HLAinteger8 | HLAunsignedInteger8 |
| HLAinteger16BE | HLAunsignedInteger16BE |
| HLAinteger16LE | HLAunsignedInteger16LE |
| HLAinteger32BE | HLAunsignedInteger32BE |
| HLAinteger32LE | HLAunsignedInteger32LE |
| HLAinteger64BE | HLAunsignedInteger64BE |
| HLAinteger64LE | HLAunsignedInteger64LE |

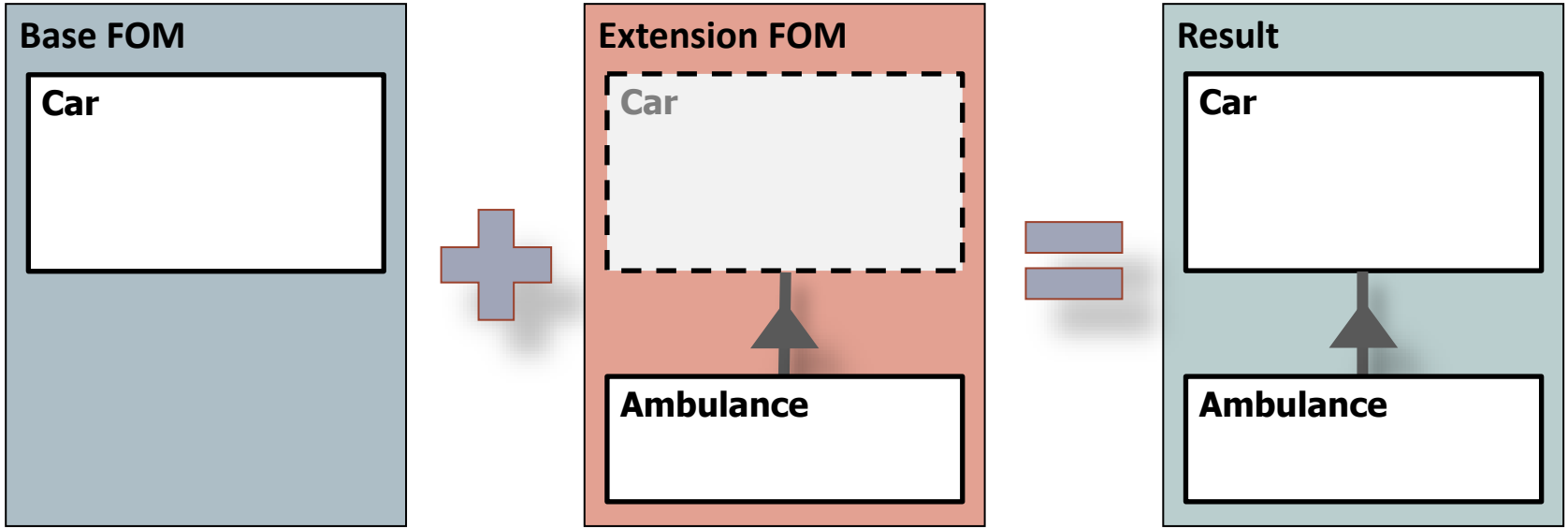
# FOM Merge – Adding Object Classes

HLA Evolved allows adding new classes.



# FOM Merge – Adding Subclasses

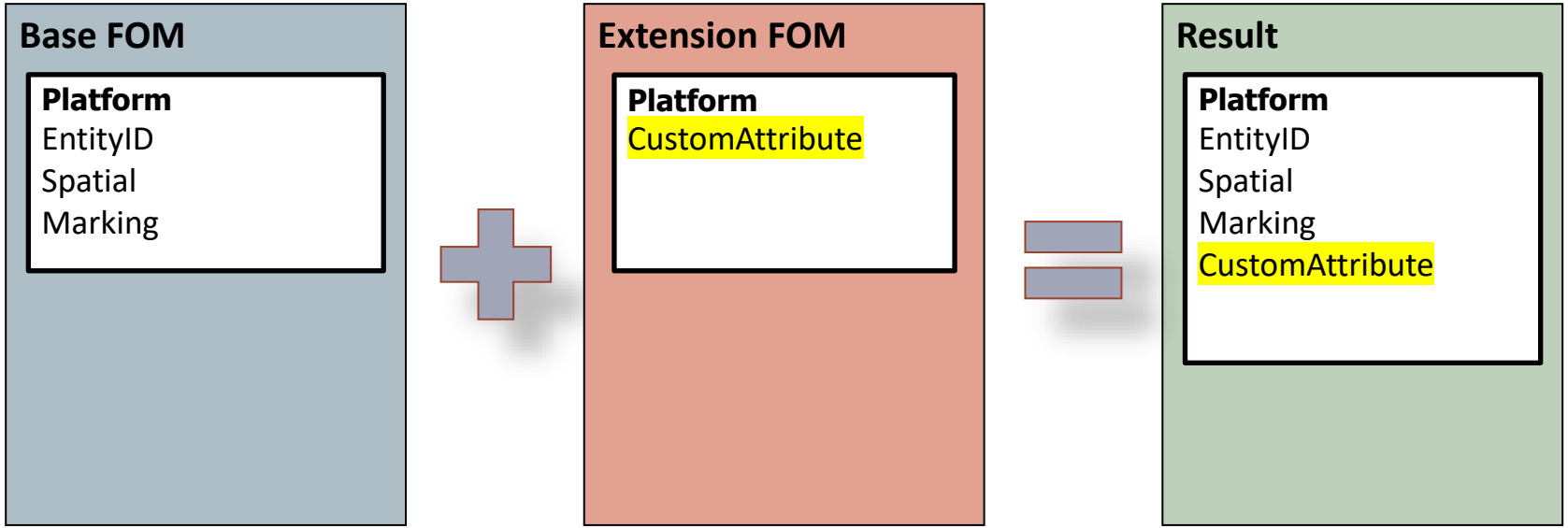
HLA Evolved allows adding new subclasses.



# FOM Merge – Add Attributes to Existing Classes

HLA 4

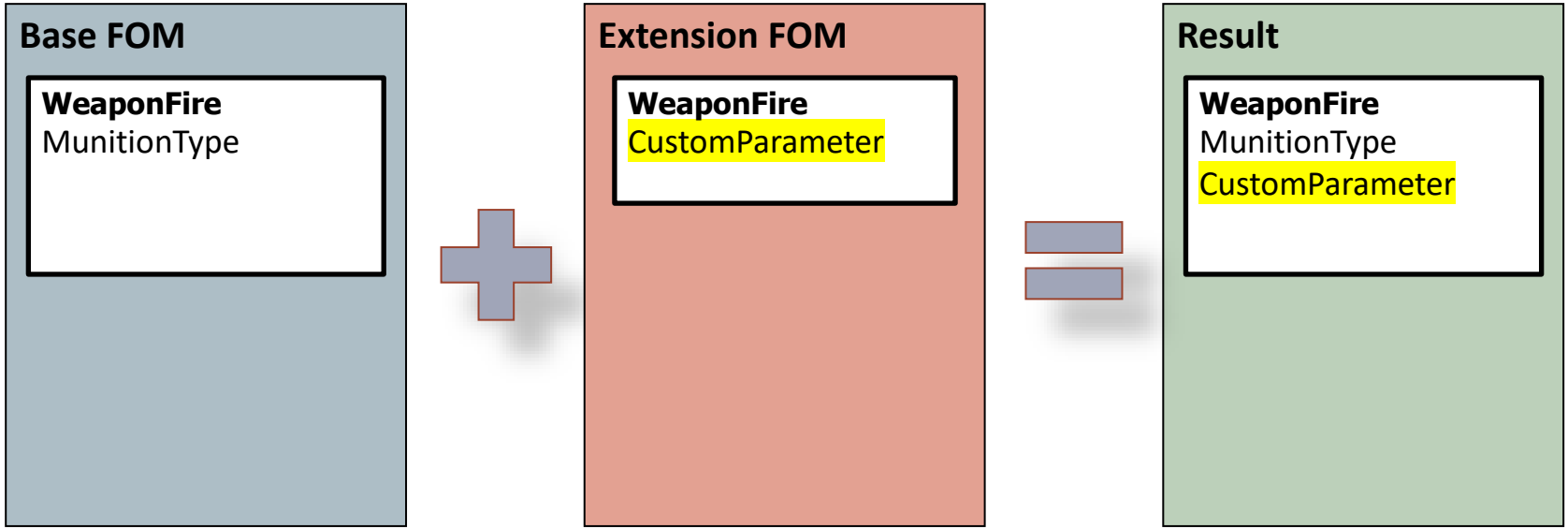
HLA 4 allows adding new attributes to existing object classes, even at runtime.



# FOM Merge – Add Parameters to Existing Interactions

HLA 4

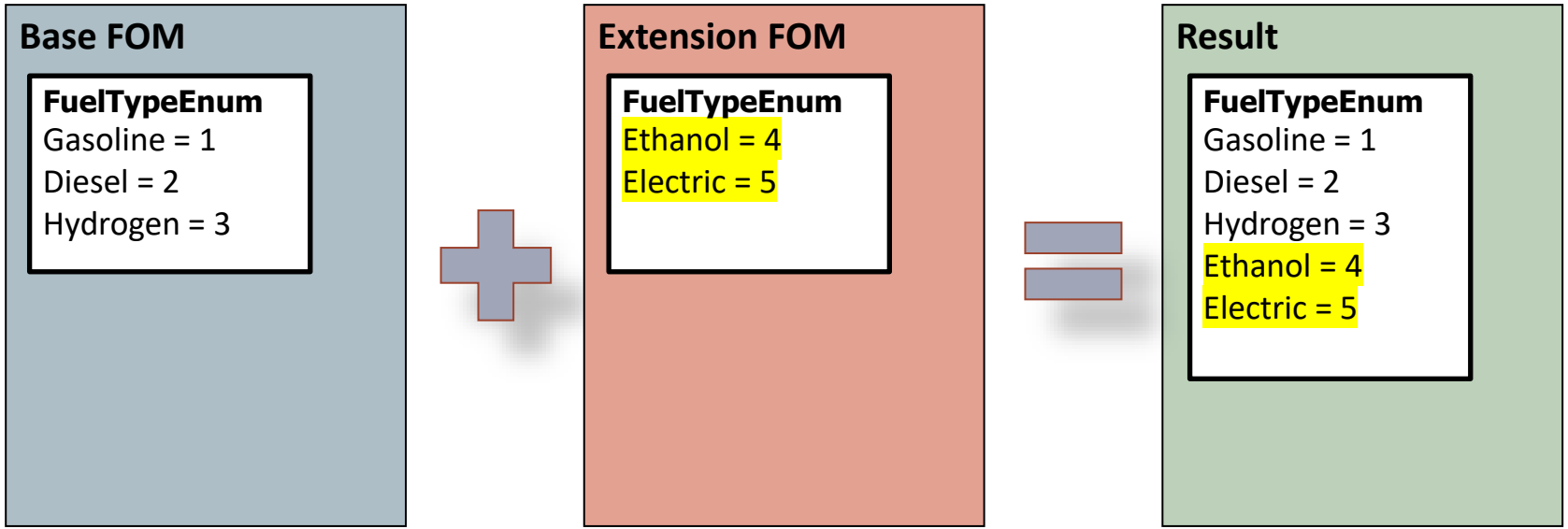
HLA 4 allows adding new parameters to existing interaction classes, even at runtime.



# FOM Merge – Extendable Enums

HLA 4

Add values to existing enums.

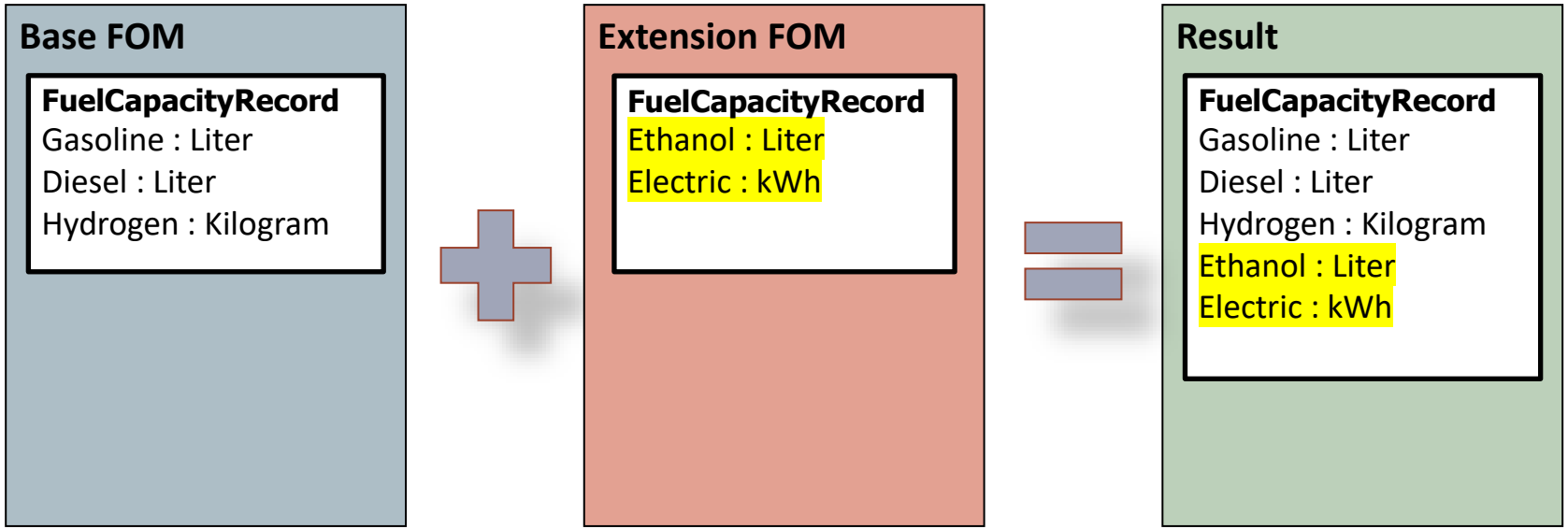


# FOM Merge – Extendable Variant Records

HLA 4

Add variants to existing variant records

New encoding `HLAextendableVariantRecord`



# Dimensions

In HLA Evolved, dimensions are specified on each attribute.

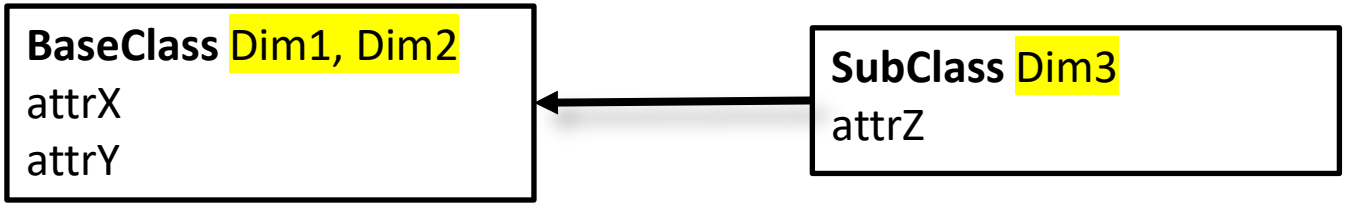

```
getAvailableDimensionsForClassAttribute(BaseClass, attrX) => ( Dim1, Dim2 )

getAvailableDimensionsForClassAttribute(BaseClass, attrY) => ( Dim2 )

getAvailableDimensionsForClassAttribute(SubClass, attrZ)  => ( Dim3 )
```

Dim3 is not available on attrX and attrY so we cannot filter them using Dim3.

# Dimensions

HLA 4

In HLA 4, dimensions are specified on the object class, and they are inherited.



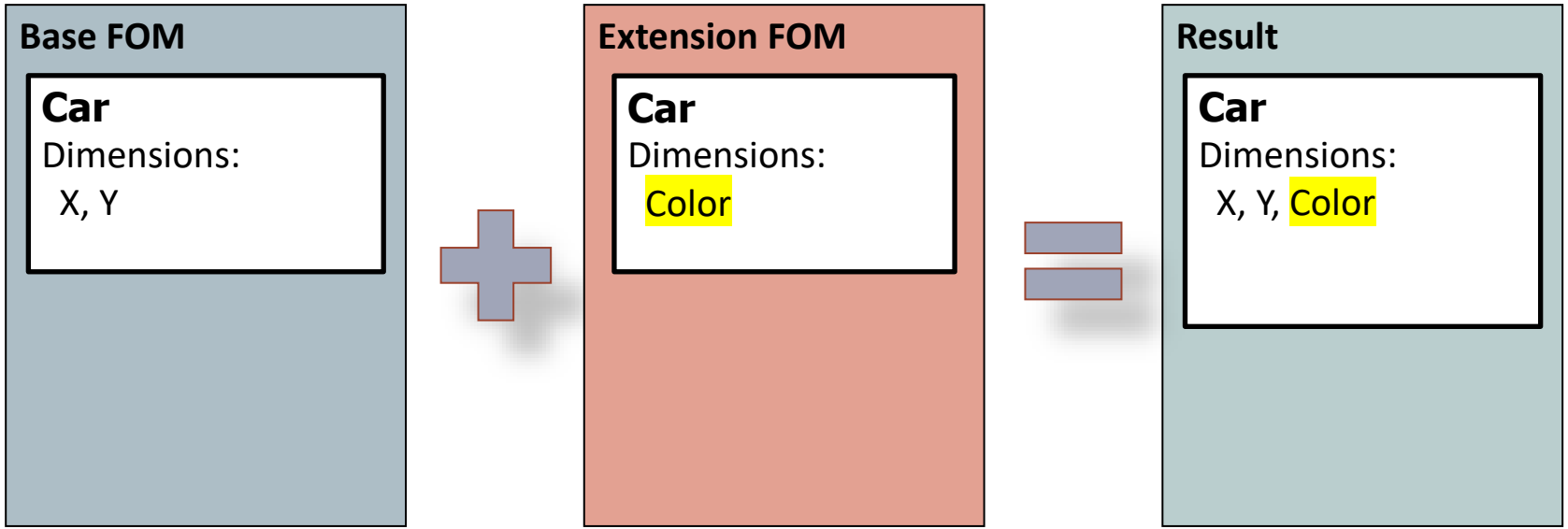

```
getAvailableDimensionsForClass(BaseClass) => ( Dim1, Dim2 )

getAvailableDimensionsForClass(SubClass)  => ( Dim1, Dim2, Dim3 )
```

- All dimensions are available for all attributes
- This makes it possible to use all the dimensions to filter all the attributes.
- For example, on a SubClass object, you can use Dim3 to filter attrX and attrY.

# FOM Merge – Add Dimensions to Object Classes

HLA 4

With HLA 4, you can add dimensions to standard FOMs.



# Required attributes

HLA 4

In HLA 4, attributes can be marked in the FOM as **Required**.

This does not affect the FDD or the behavior of the RTI.

*“any federate that registers an instance of this class shall either publish and provide a value for this attribute or allow another federate to take ownership of the attribute and provide a value”*

# Object References

In HLA Evolved, you could refer to objects using a name, a unique id, etc. But this was ad hoc.

Car

Name=”Red Car 14”
Driver=”John”
Passengers=[”Jack”, ”Jill”]

**Car**
Make : HLAunicodeString
Model : HLAunicodeString
Driver : HLAunicodeString

Person

Name=”John”

**Person**
Name : HLAunicodeString

# Object References

HLA 4

HLA 4 formalizes references using a Reference Datatype.

**Car**

Name : String = ”Car #1”
Driver : PersonReference = ”John”

**Person**

Name = ”John”

**PersonReference**
Representation: String
Referenced class: Person
Referenced attribute: Name

The **Reference Datatype** defines the type of the value, what kind of object it references, and which field of that object that shall match.

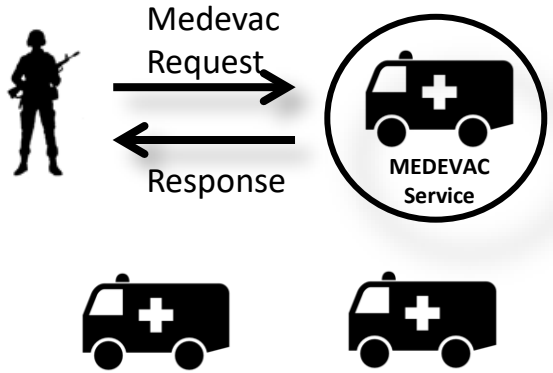
# Send Directed Interaction

HLA 4

Similar to Send Interaction but takes an object instance as parameter.

Directed interaction is delivered to all federates that own attributes on the specified object instance.

| Object | Directed Interaction | P/S |
|---|---|---|
| Ambulance | MedevacRequest | S |
| Soldier | MedevacRequest | P |




```
publishDirectedInteraction(Soldier, MedevacRequest)
subscribeDirectedInteraction(Ambulance, MedevacRequest)

sendDirectedInteraction(MedevacRequest, ambulance1, parameters, …)
receiveDirectedInteraction(MedevacRequest, ambulance1, parameters, …)
```

# Improved Fluent Encoding Helpers

HLA 4

## HLA Evolved

```
HLAinteger32BE y;
y.set(value);
auto encodedBuffer = y.encode();
```

## HLA 4

```
auto encodedBuffer = HLAinteger32BE()
   .set(value)
   .encode();
```

# Improved logging format

HLA 4

In HLA Evolved, MOM logging was a free-form, implementation-dependent text format.

In HLA 4, MOM logging uses a documented json-like format.

```
[
  [
    "AttributeHandle<1001>",
    "xFDcMa"
  ],
  [
    "AttributeHandle<1002>",
    "R6lkHla"
  ]
]
```

# Improved Switches

HLA 4

In HLA Evolved, switches were managed by a confusing mix of FOM settings, API calls, and MOM attributes and interactions.

In HLA 4, switch handling is unified.

- All switches are available in the FOM
- All switches have get (and set) methods in the API
- All switches can be accessed through the MOM

# Summary of FOM Changes

HLA 4

- Unsigned Integers
- Add attributes to existing classes
- Add parameters to existing interaction classes
- Extendable enums and variant records
- Dimensions defined per class, not per attribute
- Dimensions are now inherited
- Add dimensions to existing classes and interactions
- Required attributes
- Object references
- Directed Interactions (to owner of attributes)
- Improved Encoding Helpers
- Improved MOM logging
- Improved switches

(Slide based on “Blank”)

Simulation Interoperability Standards Organization

"Simulation Interoperability & Reuse through Standards"

# Q&A / Discussion