



OneSAF Interoperability Mechanisms

August 2004



General Interoperability Requirements

- **4.13 Interoperability**

OneSAF will be interoperable with constructive, virtual, and live simulators and simulations such that a simulation environment is created so that no object in any one simulation enjoys an advantage or disadvantage due to the simulation in which it resides (e.g., a fair fight).



Constructive Simulation Interoperability Requirements

- **4.13.1 Constructive Simulations**

OneSAF will be interoperable and have the ability to connect with constructive simulations. OneSAF must interoperate with the version of ACTS fielded at the time of OneSAF FOC as defined in DOD 5000.59-M, DoD M&S Glossary, and be capable of the following specific interactions:

- a. Correlation of terrain models to ensure common location across simulations;**
- b. Arbitrate differing levels of resolution to ensure a level playing field (also known as "fair fight");**
- c. Negotiate resolution of attrition and kill removal between dissimilar entities;**
- d. Resolve LOS between entities in the separate simulations in a realistic manner;**
- e. Migrate individual entities across the simulation boundaries (move from OneSAF to ACTS and vice versa);**
- f. Arbitrate aggregation/disaggregation between the two simulations to maintain the level playing field;**
- g. Correlate data bases to correct for differences in physical measurement and methods for quantification of database elements; and**
- h. Resolve differences in the fidelity of individual models between the two simulations.**



ACTS Interaction Interoperability Requirements

- **4.13.1 Constructive Simulations**

... Operators will see no difference in stimulation and interaction with units portrayed in either ACTS or OneSAF.

OneSAF will have the capability to represent aggregate units in ACTS, (or in special cases as individual platforms) as a unit decomposed into individual platforms and command entities.

- **For example, a platoon that is represented in OneSAF will have the same representation of type and quantity of weapon systems (to include lethality and vulnerability characteristics), personnel, supplies (Class II, IV, and V), condition (alive, mobility-kill, etc.), as the platoon represented in the constructive simulation (ACTS).**

In addition, minefields and breaches to minefields will have a consistent representation in both ACTS and OneSAF.

Also included is the consistent portrayal of weather and battlefield phenomenology.

A BLUFOR/OPFOR unit in OneSAF will be able to detect, engage, and attrit an OPFOR/BLUFOR unit in ACTS.

OneSAF must accept Situation Awareness (SA) passed from ACTS.

Where appropriate a unit in OneSAF will provide to, and accept from, a unit in ACTS all classes of supply and personnel.



Virtual Simulation Interoperability Requirements

- **4.13.2 Virtual Simulators**

OneSAF will have the capability necessary to retire CCTT and AVCATT SAF and be used by the CCTT and AVCATT manned modules.

Units in OneSAF will have the capability to interact with CCTT/AVCATT manned modules.

- **For example, a OPFOR unit in OneSAF will be able to detect, engage, and attrit a CCTT/AVCATT manned module with sufficient fidelity to ensure a fair fight**
- **Where appropriate, a BLUFOR unit in OneSAF will be able to provide to, and accept from, a CCTT/AVCATT manned module all appropriate classes of supply and personnel;**
 - **e.g., a cargo aircraft could move troops and ammo from point A to B in AVCATT and provide it to a unit in OneSAF.**



Virtual Simulation Interoperability Requirements (cont.)

- **4.13.2 Virtual Simulators**

...Physical characteristics of the Battlespace, for all Army Universal Task List (AUTL) categories, will be consistently portrayed for both CCTT/AVCATT manned modules and OneSAF.

- **For example, minefields and breeches to minefields will have a consistent representation in CCTT/AVCATT and OneSAF.**
- **Also included is the consistent portrayal of weather and battlefield phenomenology.**

CCTT/AVCATT manned modules and OneSAF will have equal fidelity in the stimulation of C4I systems.

Operators will see no difference in stimulation and interaction with units portrayed in either CCTT/AVCATT or OneSAF.

OneSAF computer generated forces will be able to ‘tag’ onto the virtual simulators and ‘follow the leader’ for coordinated group execution of combat missions

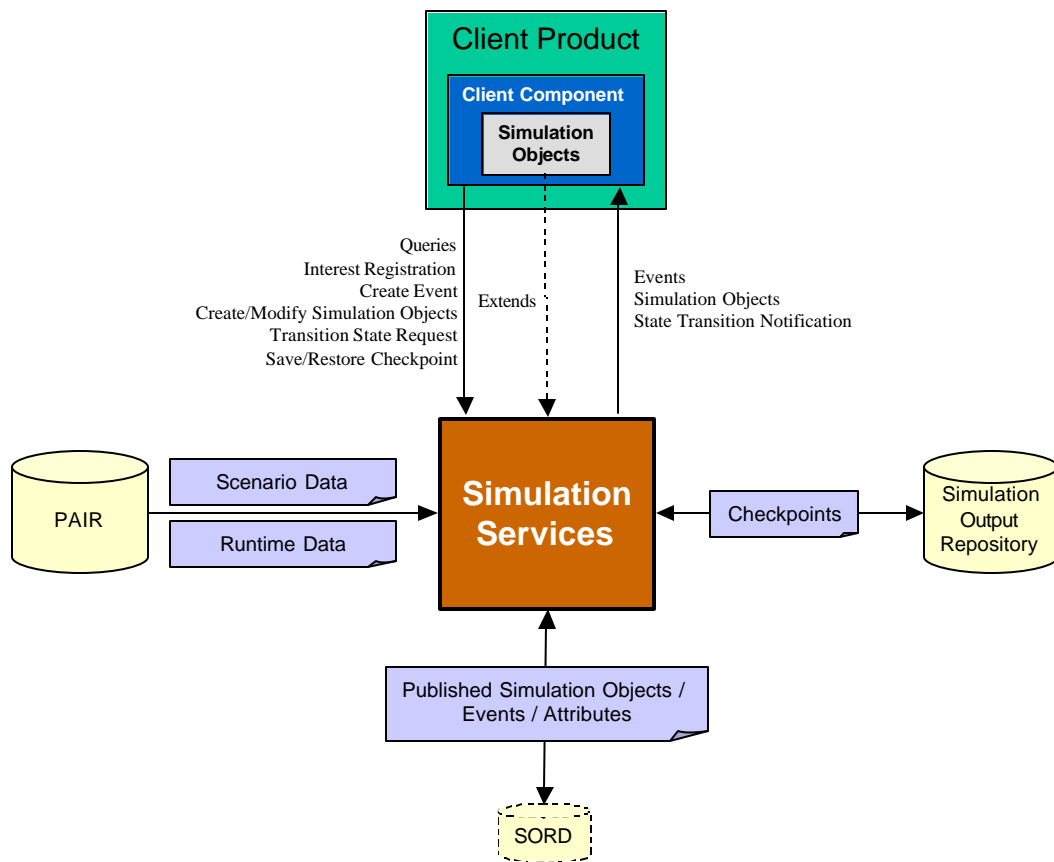
- **(i.e. formation flying, battle position occupation, attack by fire position occupation, and engagement area direct fire operations.)**



OneSAF Simulation Engine Capability

- **OneSAF distributed simulation, including interoperability with other simulations and live systems, is implemented using the OneSAF Simulation Services component within the OneSAF Support Layer**

- **Provides a Simulation Engine for OneSAF including time and event management**
- **Provides RTI and DIS interface services**
- **Provides standalone and distributed simulation capabilities**
- **Supports simulation checkpoint**



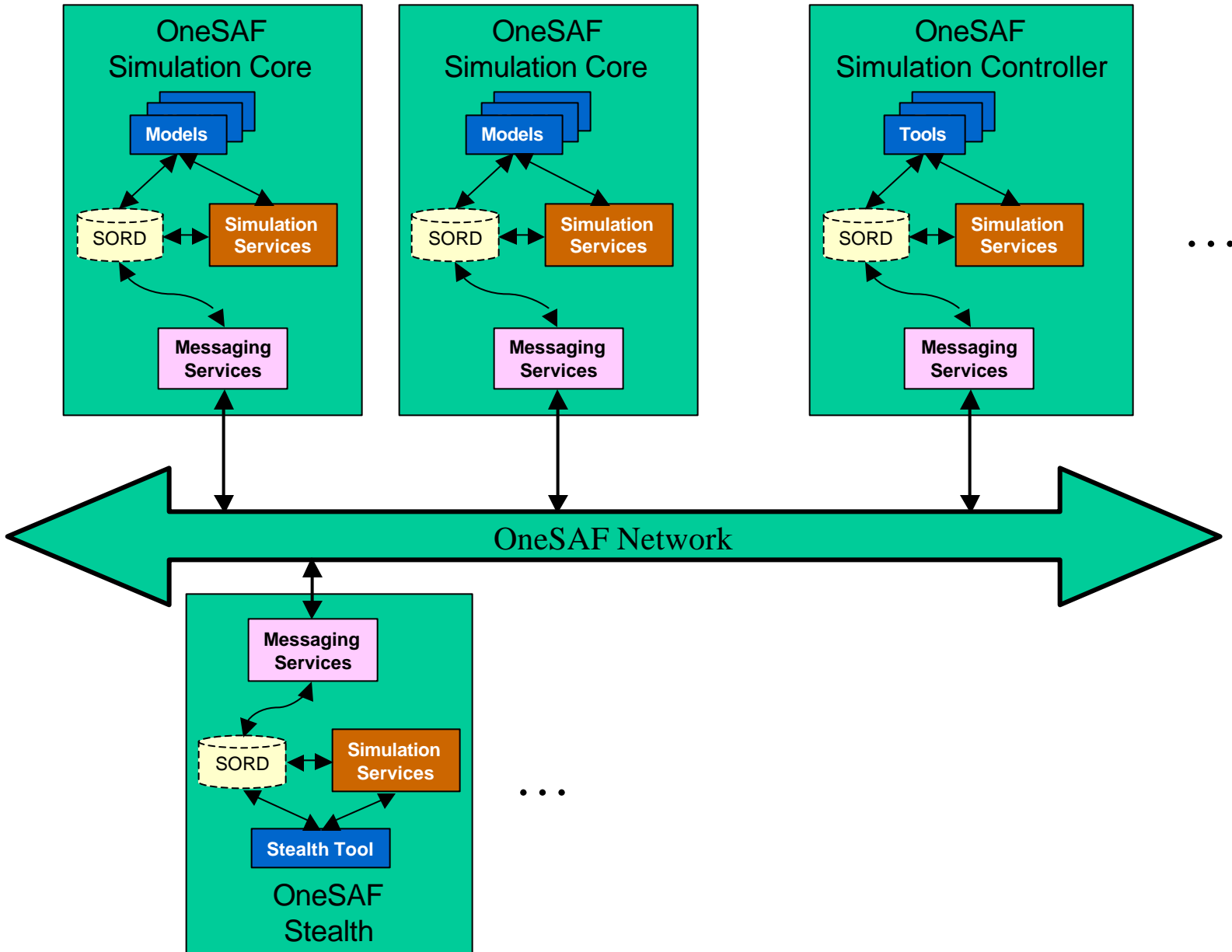


OneSAF Simulation Object Runtime Database (SORD)

- **The SORD provides the transparent network distribution capabilities for OneSAF**
 - **A virtual database to support distributed simulation**
 - **Exists in-memory during the event phase**
 - **Maintained by the Simulation Services infrastructure**
- **The SORD contains shared**
 - **Battlespace simulation objects**
 - **Platforms, units, dynamic environment objects (smoke clouds, obstacles) missions, orders, and reports**
 - **Simulation control objects**
 - **Allows tools and other applications to participate in the simulation infrastructure and interact with battlespace simulation objects**
- **The SORD supports interest management**
 - **Each SORD client can access the data in the database if it has properly expressed interest with the infrastructure for those objects**
- **The SORD supports runtime access to simulation objects**
 - **Persistent over time, supporting query, notification, modification, and deletion**
- **The SORD supports runtime access to simulation events**
 - **Transitory, and made available to interested clients when processed according to their scheduled event times**



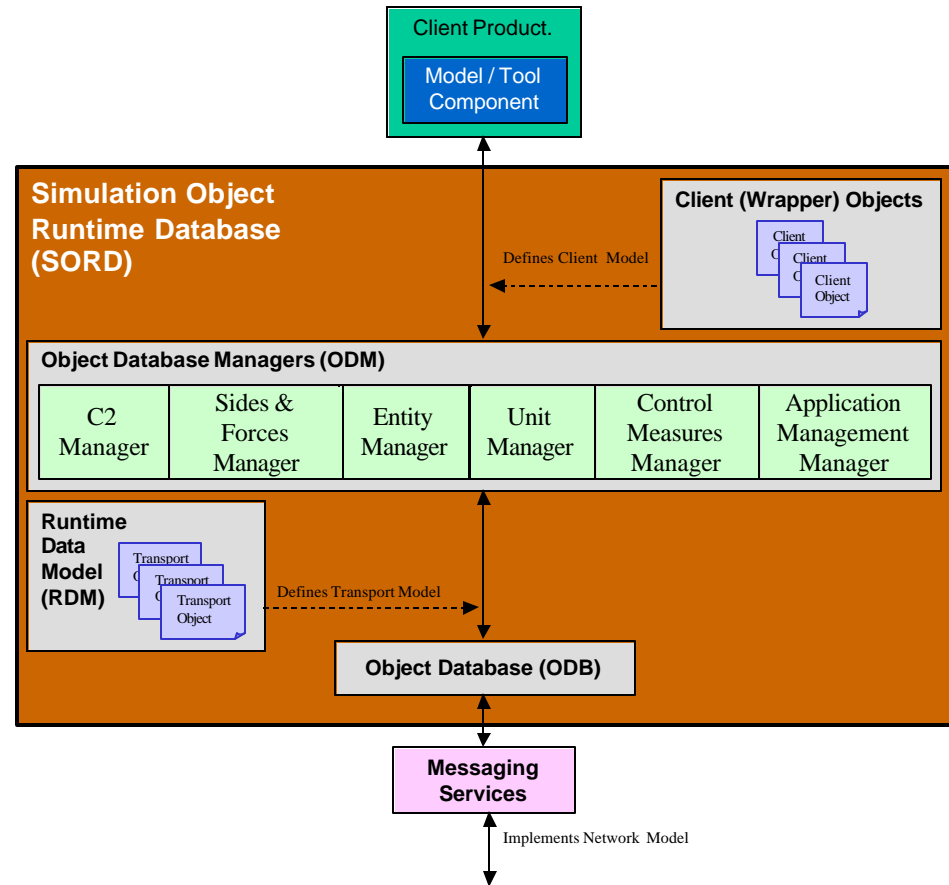
OneSAF Distributed Simulation





OneSAF SORD Implementation

- The SORD is implemented in three parts
 - The Object Database (ODB) provides the low level capability to distribute objects and interactions across the network
 - The Runtime Data Model (RDM) defines the network objects and interactions used by OneSAF to represent and control the battlespace
 - The Object Database Manager (ODM) layer contains individual managers that represent rich objects and events used by tool and modeling clients
 - These managers map the rich objects and events into RDM objects and interactions that are distributed through the ODB
- The SORD relies on OneSAF Messaging Services for efficient network delivery of distributed data





Purpose of the Object Database

- **The Object Database (ODB) translates the Runtime Data Model (RDM) into a Network Transport model**
 - **Runtime Data Model describes publicly available information that flows around a OneSAF System instance**
 - **Network Model describes the physical packaging of that information to transport it between nodes, across the network**
- **The Object Database supports persisting RDM information in two ways:**
 - **Short term, binary form for temporary checkpoints**
 - **Long term, XML based form for**
 - **Maintaining between versions of the system**
 - **Sharing with external clients of OneSAF**
- **Provide a pure Java collection of objects to communicate state between simulation nodes**



OneSAF ODB Implementation

- **Supports objects and interactions**
 - **Objects**
 - Define the state of something
 - Persist over time
 - Can refer to other objects
 - Cannot refer to interactions
 - **Interactions**
 - Convey an action
 - Are not persisted
 - Can refer to ODB objects
 - <Cannot refer to other interactions>
- **Implemented as a Java Collection containing the objects stored in the ODB**
- **Requires clients of the ODB to register the types (interface classes) of objects and interactions intended to be supported by the ODB**
- **Contains methods to**
 - Create an instance of an object or interaction
 - Add a created object to the collection so that it will be distributed
 - Fire a created interaction so that it will be distributed
- **Uses Java Dynamic Proxies to implement ODB objects or interactions when created**
 - The proxies intercept set and get methods in the ODB objects to effect distributed object synchronization

Events Associated with ODB Objects/Interactions

- **Can listen for**
 - **objectCreated**
 - Notification for when objects are created
 - Sent to notify nodes of a new object added to the ODB
 - Globally synchronous
 - **objectRemoved**
 - Notification of when objects are removed
 - Sent to notify when and object is deleted from the ODB
 - Globally synchronous
 - **propertyChanged (object level)**
 - Sent to notify that an attribute (or multiple attributes) have been changed in an object
 - Locally synchronous
 - **interactionFired**
 - Sent to notify that an interaction has been fired (Interaction is the event source)
 - Globally synchronous



ODB Features

- **The ODB**
 - **Preserves referential integrity between objects**
 - Cannot delete an object that is pointed to by another object
 - References can only be one level deep
 - Objects cannot point to interactions
 - **Presents a type safe, Java interface to clients instead of a bag of bytes**
 - **Supports Transient Objects**
 - Transients do not get persisted to disk in a save
 - Transients cannot be referred to by other objects
 - ApplicationNode object and AllocationObjects are examples
 - **Supports Periodic Types**
 - Periodic types are not sent reliably
 - Periodic types guarantee they will be updated at a frequent rate
 - Consistency of Periodic Types is maintained by sending the entire object for each attribute update (normally the ODB just sends the attribute)
 - **Supports registration of dynamically created types, called synthetic types**



ODB Performance

- **ODB attribute update send: 0.7 milliseconds**
- **ODB attribute update round trip (between multiple nodes) time 5-7 milliseconds heavily loaded**
 - **A big driver here appears to be thread switching latencies in the OS kernel**

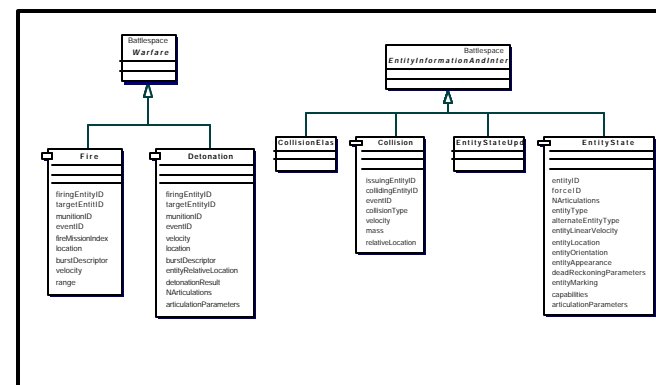
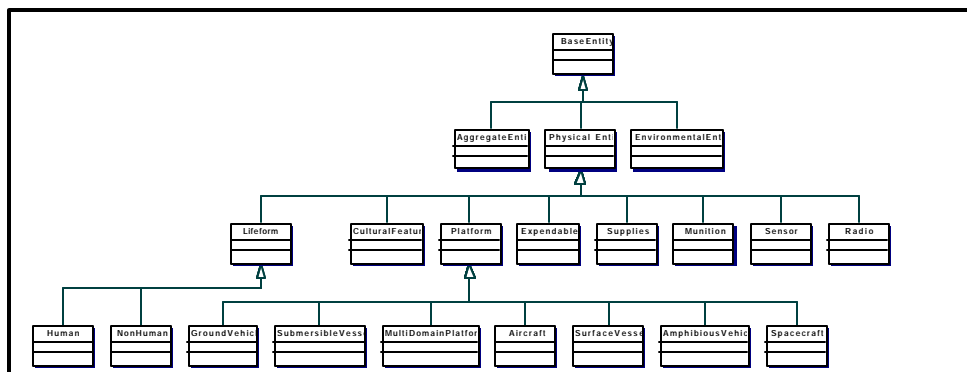


Runtime Data Model (RDM)

- **“The runtime data describes what simulation objects and events can be created and distributed by the simulation engine. In addition, the runtime data specifies which attributes are valid for each simulation object and event.”**
 - **3.5.2 PLAS Volume II**
- **Significance**
 - **The RDM determines what objects and events are in the battlespace, as well as the objects and events (interactions) that control or manage the battlespace**
 - **It establishes our effective Simulation Object Model (SOM), in HLA parlance**
 - **The RDM is the basis for internal model development**
 - **The RDM defines what simulated objects and interactions are available**
 - **Models are ultimately implemented in terms of these objects and interactions**
 - **Update and reaction to changes in object state**
 - **Generation and processing of interactions**
 - **The RDM is the basis for interoperability with external systems**
 - **Simulations, C4I devices, live devices, etc.**

OneSAF RDM Implementation

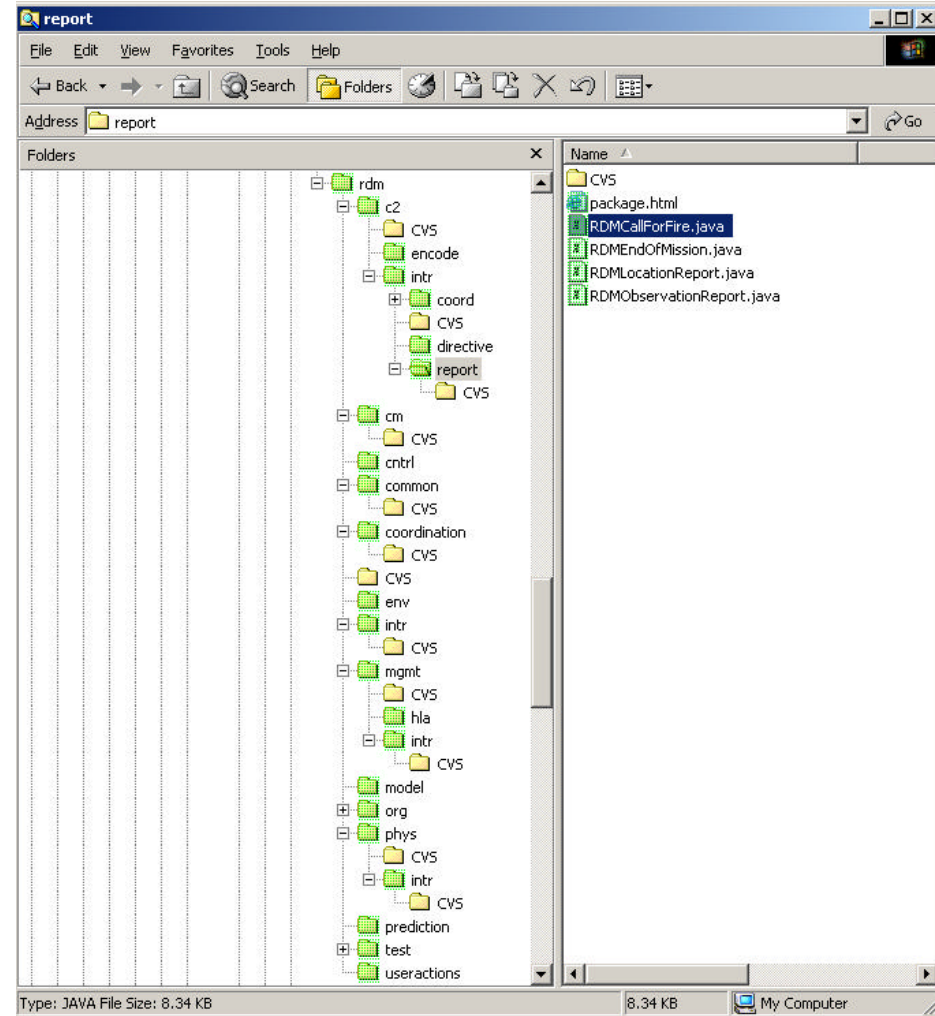
- Based on analysis, RPR FOM / DIS is the most mature and richest entity-based data model generally available and published
 - More than a decade of data engineering founded on DIS-style interoperability
- Though RPR-FOM is a good fit, OneSAF has specific requirements that necessitate extension
 - I.e., multi-sided relationships
- Though RPR-FOM is a good fit, it is incomplete
 - Control Measures
 - Command and Control (Orders, Reports)
 - Environmental Effects
 - OOS-specific Application Management
- OneSAF approach:
 - Base OOS RDM on RPR-FOM
 - Extend as necessary, concentrating on incomplete areas
 - Keep abreast of other FOMs as input





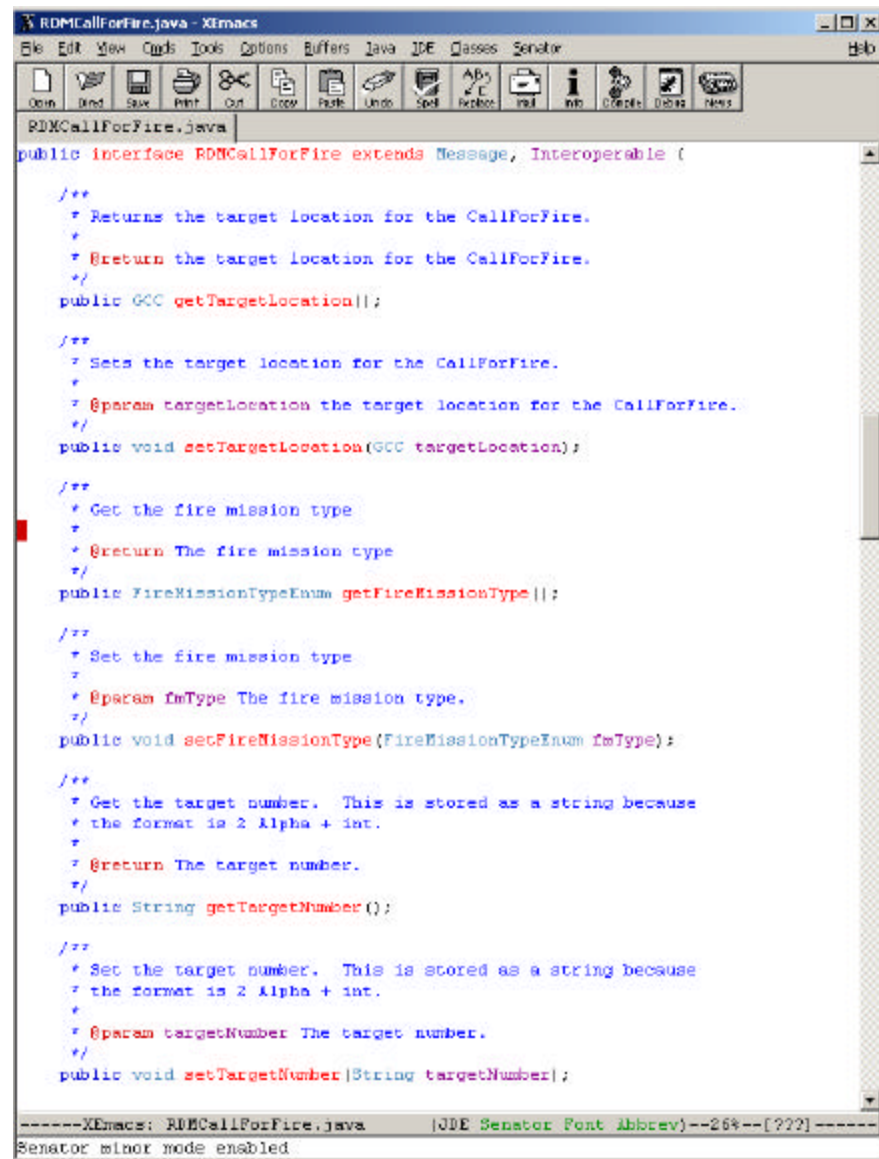
OneSAF RDM Structure

- The RDM is organized according to type of object or interaction
 - **C2: Command and Control**
 - **CM: Control Measure**
 - **Cntrl: Control interactions**
 - **Common: Common types**
 - **Coordination: Coordination interactions**
 - **Env: Environmental objects and interactions**
 - **Intr: General interactions**
 - **Mgmt: Application management objects and interactions**
 - **Org: Organizational objects (units, sides, forces)**
 - **Phys: Physical objects and interactions (entities, fire, detonation, etc.)**
 - **Prediction: Predictive contracts**
 - **Test: Test objects and interactions**
 - **User: Scriptable user events (I.e., create entity)**



OneSAF RDM Contents

- Each RDM Object or Interaction is defined according to the rules of the ODB
 - Defined as an interface class
 - Containing attributes that are
 - Primitives
 - Attributable types (Encodable / Cloneable / Externalizable)
 - References to other Objects
 - Arrays of any of the above
 - Documented for purposes of interoperability
 - In the future, will be published in a more human readable format
- The domain objects represent the internal object model for OneSAF



```
RDMCallForFire.java - XMacros
File Edit View Cmds Tools Options Buffers Java IDE Classes Senator
Open Find Save Print Cut Copy Paste Undo Spell Editor Mail Info Create Delete Help
RDMCallForFire.java
public interface RDMCallForFire extends Message, Interoperable {
    /**
     * Returns the target location for the CallForFire.
     *
     * @return The target location for the CallForFire.
     */
    public GCC getTargetLocation();

    /**
     * Sets the target location for the CallForFire.
     *
     * @param targetLocation the target location for the CallForFire.
     */
    public void setTargetLocation(GCC targetLocation);

    /**
     * Get the fire mission type
     *
     * @return The fire mission type
     */
    public FireMissionTypeEnum getFireMissionType();

    /**
     * Set the fire mission type
     *
     * @param fmType The fire mission type.
     */
    public void setFireMissionType(FireMissionTypeEnum fmType);

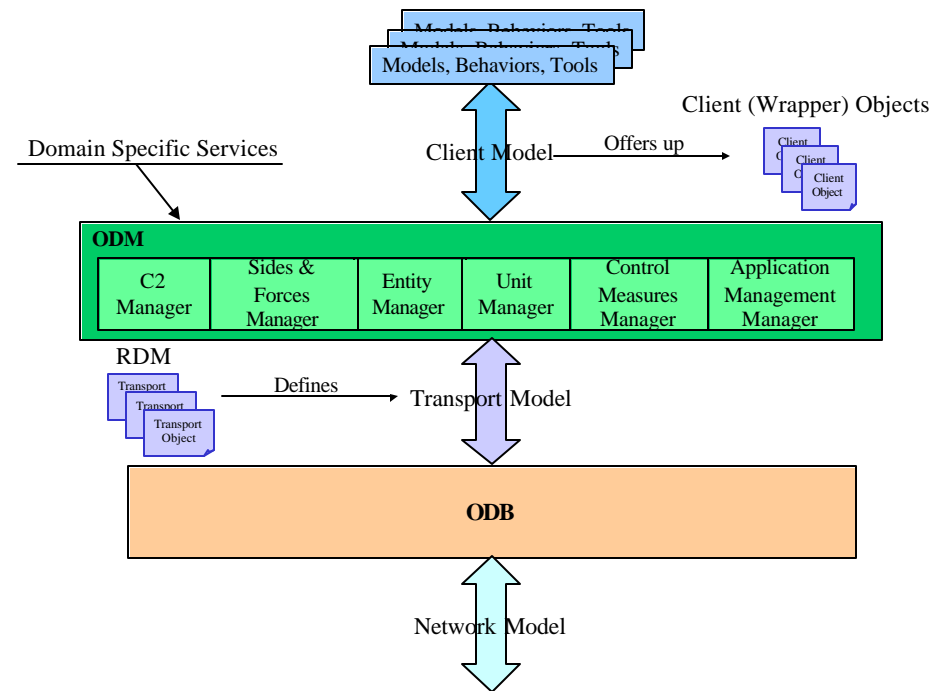
    /**
     * Get the target number. This is stored as a string because
     * the format is 2 Alpha + int.
     *
     * @return The target number.
     */
    public String getTargetNumber();

    /**
     * Set the target number. This is stored as a string because
     * the format is 2 Alpha + int.
     *
     * @param targetNumber The target number.
     */
    public void setTargetNumber(String targetNumber);
}
-----XMacros: RDMCallForFire.java |JDE Senator Font Abbrev|--26%--[???]-----
Senator minor mode enabled
```



Object Database Managers (ODMs)

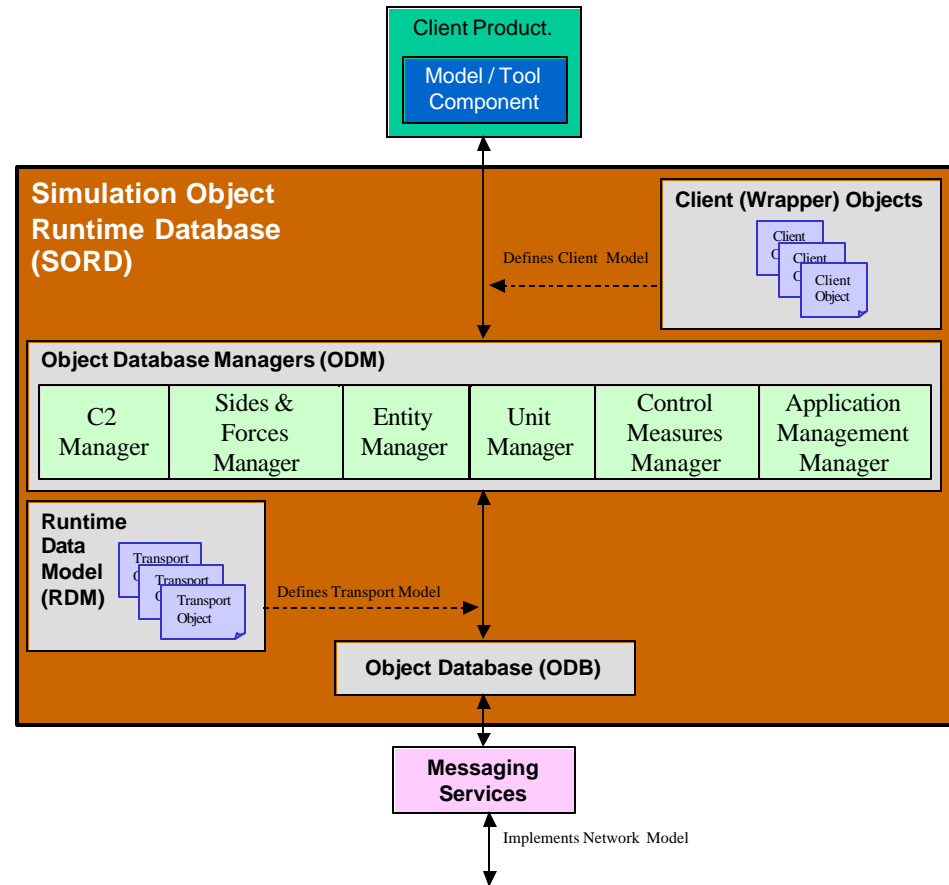
- Managers provide a layer between applications (clients) and raw transport objects (RDM objects)
 - Create
 - Destroy
 - Collection
 - Including Area of Interest (AOI) binning
- For abstraction, managers provide client (wrapper) objects
 - Wrappers intercept sets and gets and translates them to sets and gets on the transport objects, with appropriate packing and mapping





Interfacing to OneSAF

- Within OneSAF, there are three ways a component can tap into the runtime distribution
 - Use Messaging Services to tap into the network model
 - Use the ODB and the RDM to tap into the data-only transport model
 - Use the ODMs to tap into the internal rich objects and services provided by the client model
- Higher levels of interface provide higher levels of service and closer coupling





HLA Interoperability Requirements

- **4 Capabilities Required**

- ... OneSAF will be a composable, HLA compliant simulation that permits the user to easily change the scenario, environment, physical models and/or data, and combat behaviors in setting up an application and analyze causes of simulation outcomes.

- **4.9 Core Physical Models**

- ...OneSAF must be able to link with higher fidelity and higher resolution models via HLA allowing substitution of resident OneSAF physical models.

- **4.14.2.2 AAR**

- OneSAF must provide the capability to be linked via HLA to the future AAR systems and have the minimum capability that exists in JANUS plus a stealth capability.

- **4.14.8.c Standalone**

- ... OneSAF will be developed under HLA and utilize the services of the HLA Run Time Infrastructure (RTI) for dynamic information exchange with other simulations.

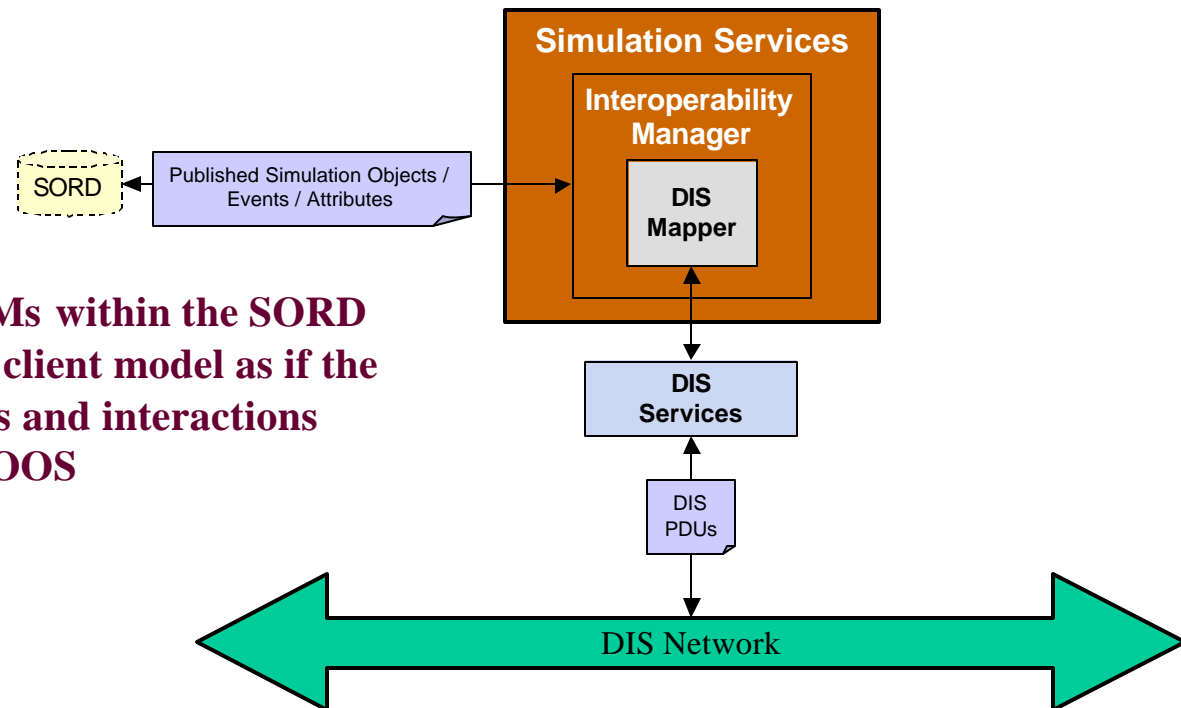


OneSAF Interoperability Management

- In OneSAF, the Interoperability Manager subcomponent within the Simulation Services taps into the ODB portion of the SORD to translate between the OneSAF Runtime Data Model (RDM) and external data models

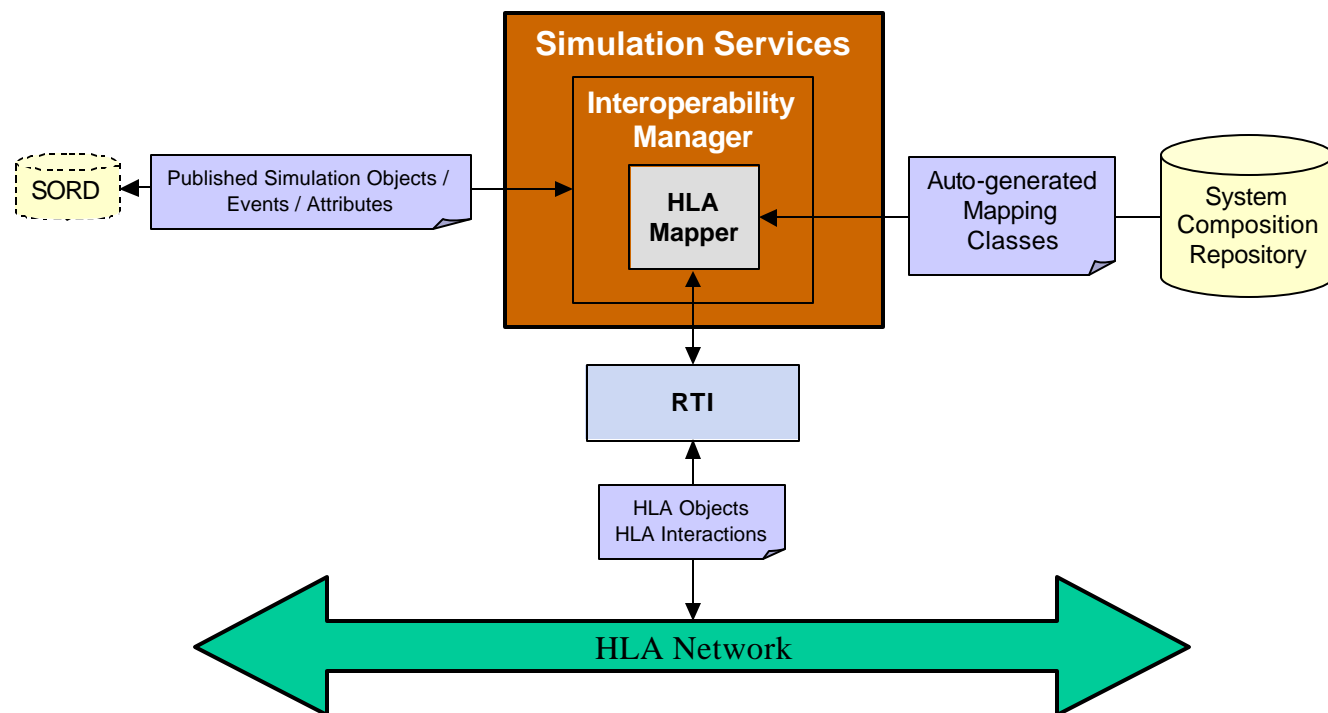
- Provides translation at the transport layer

- Allows the ODMs within the SORD to flesh out the client model as if the external objects and interactions were native to OOS



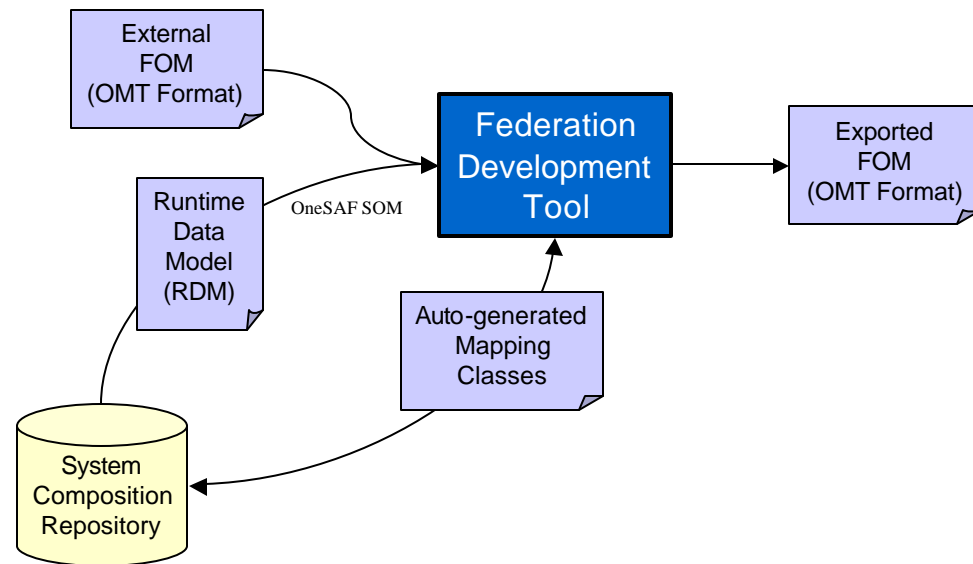
OneSAF Interoperability Management (cont.)

- Current instances of the Interoperability Manager provide gateway capabilities between OneSAF and HLA, and between OneSAF and DIS
- Future instances will provide gateway capabilities to CTIA, TENA, FCS, etc.



HLA Interoperability Management

- Because HLA supports multiple data models as defined by the particular Federation Object Model (FOM) used in a federation, OneSAF provides a GUI-based Federation Development Tool (FDT) to allow users to specify federation-specific mappings between the OneSAF RDM and a particular federation's FOM
- FDT uses code generation to construct the software mappings based on user inputs
- FDT supports the generation of an Object Model Template (OMT) formatted description of the internal OneSAF Simulation Object Model (SOM) (I.e., the RDM)
 - Provides a standard mechanism to inform other federates of OneSAF's SOM (I.e., its FOM preference)
- FDT supports the generation of an OMT description of the actual FOM OneSAF intends to map to, based on the user-specified mappings for this federation
 - Provides a standard mechanism to share the FOM with other federates





Interoperating with OneSAF

- **Other simulations can interoperate with OneSAF by:**
 - **Interfacing internally via any one of the 3 interfacing levels**
 - Network level
 - Transport level
 - Client level
 - **Interfacing externally using simulation interoperability mechanisms**
 - HLA
 - DIS
 - **Interfacing externally using other interoperability mechanisms supported by OneSAF**
 - In the future, CTIA, TENA, FCS, etc.



Live Interoperability Requirements

- **4.13.3 Live**

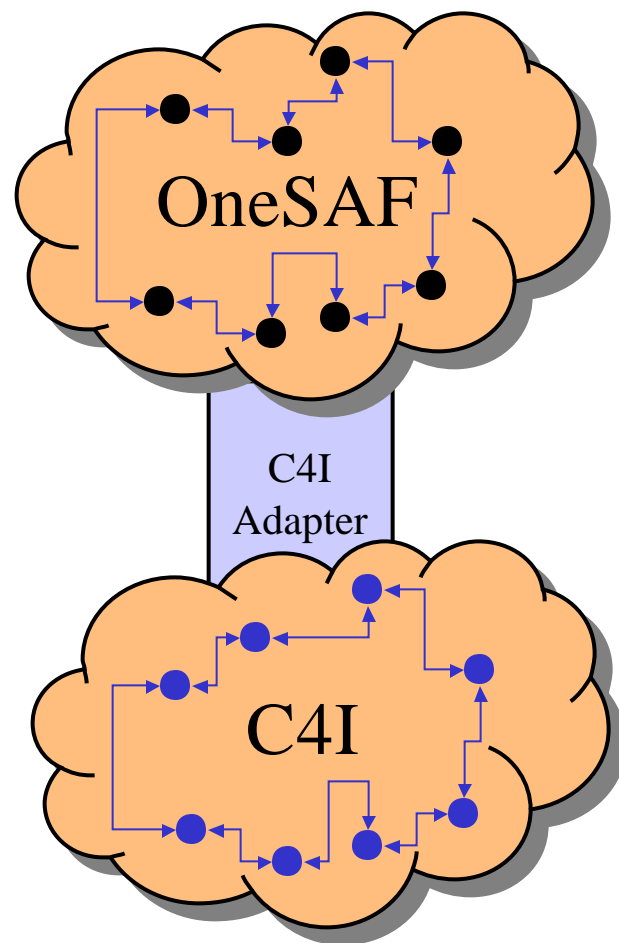
OneSAF will stimulate and be stimulated by Army C4I systems and will support real-time mission execution.

- **At the tactical level, ABCS includes both the ATCCS and FBCB2.**
- **OneSAF will represent and stimulate both ATCCS and FBCB2 in a manner that emulates the real world information flow between vertical echelons of command and horizontally across echelons.**



C4I Interoperability

- **The OneSAF C4I Adapter provides interoperability between OneSAF and real-world C4I devices**
- **Translates orders and reports between C4I-native mechanisms and command and control objects in the RDM**
 - **Orders**
 - **Reports**
 - **Missions**
 - **Control measures**
 - **Etc.**
- **Translation is two way**
 - **OneSAF RDM objects to C4I orders and reports**
 - **C4I orders and reports to OneSAF RDM objects**
- **Once a C4I order or report is translated to an RDM-level order or report, it can be distributed within OneSAF and picked up by the appropriate C2 Managers at the ODM level**
 - **Models or tools will react to these as if they were natively generated**





Future (P3I) Interoperability Requirements

- **6.9.1 Virtual Simulators.**
 - OneSAF will have the capability to be used by the SE Core (CATT Family) manned modules.
 - Units in OneSAF will have the capability to interact with SE Core manned modules. For example, an OPFOR unit in OneSAF will be able to detect, engage, and attrit a SE Core manned module with sufficient fidelity to ensure a fair fight. Where appropriate, a BLUFOR unit in OneSAF will be able to provide to, and accept from, a SE Core manned module all appropriate classes of supply and personnel; e.g., a cargo aircraft could move troops and ammo from point A to B in AVCATT and provide it to a unit in OneSAF.
 - Physical characteristics of the battlespace, for all AUTL categories, will be consistently portrayed for both SE Core manned modules and OneSAF. For example, minefields and breeches to minefields will have a consistent representation in CATT and OneSAF. Also included is the consistent portrayal of weather and battlefield phenomenology.
 - SE Core manned modules and OneSAF will have equal fidelity in the stimulation of C4I systems. Operators will see no difference in stimulation and interaction with units portrayed in either CATT or OneSAF. OneSAF computer generated forces will be able to ‘tag’ onto the virtual simulators and ‘follow the leader’ for coordinated group execution of combat missions (i.e. formation flying, battle position occupation, attack by fire position occupation, and engagement area direct fire operations.)



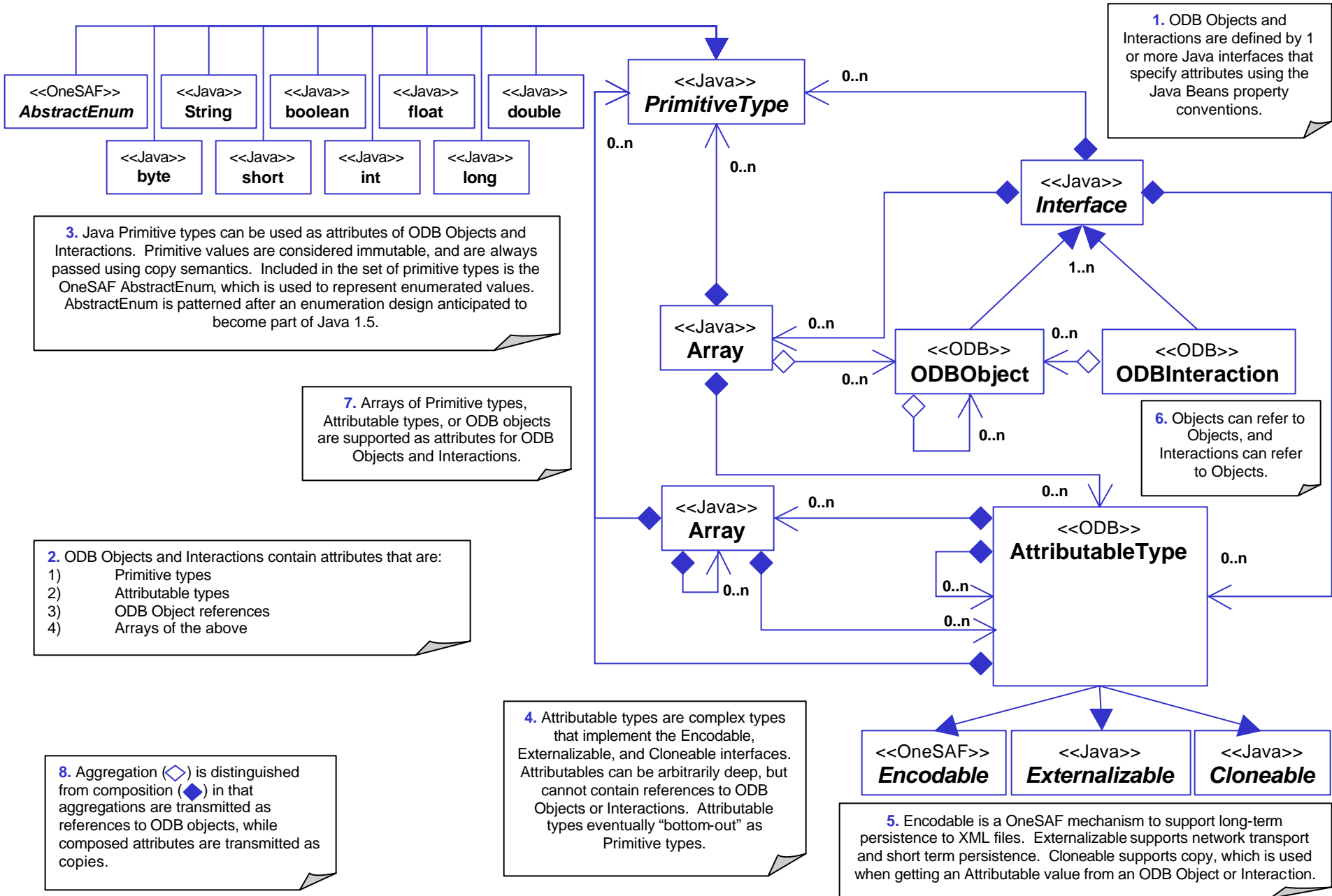
Future (P3I) Interoperability Requirements (cont.)

- **6.9.2 Live.**
 - **OneSAF will be directly linked to real-world C4I systems and will support real-time mission execution. OneSAF will be interoperable with ABCS. OneSAF must link with live and real-time casualty assessment systems. OneSAF will accept orders from the real world C4I systems (analog and digital) so that it is transparent to the C4I system operator and simulation users.**
- **6.9.3 Other C4I Systems.**
 - **OneSAF will be capable of accessing information derived from mission planning in the Air Force Mission Support System (AFMSS) and the derivative Aviation Mission Planning System (AMPS). OneSAF must model the capabilities of Army Airspace Command and Control (A2C2).**
- **6.9.4 Combat Training Center (CTC) Objective Instrumentation System (OIS) Interoperability and Home Station Instrumentation System (HSIS) Interoperability**
 - **OneSAF will be interoperable with live entities such as Combat Training Center/Home Station Instrumentation System (CTC/HSIS).**



Backups

ODB Meta-Model





ODB Object Example

```
package net.onesaf.services.sys.odt.tester;  
  
public interface Line {  
  
    String getName();  
  
    void setName(String name);  
  
    Location[] getPoints();  
  
    Location getPoints(int index);  
  
    void setPoints(Location[] points);  
  
    void setPoints(int index, Location point);  
  
    void setNextLine(Line nextLine);  
  
    Line getNextLine();  
  
}
```

- **Objects are essentially collections of attributes (data)**
- **Defined as an Interface**
 - No concrete implementation specified
 - No behavior specified
- **Attributes identified by JavaBeans Interface (getters and setters only)**
- **No support for read-only attributes**
 - For each getter, there must be an equivalent setter
- **Though not shown in this example, the objects should contain javadoc comments so that the definition of the object can be published for interoperability**



ODB Object Constraints

```
package net.onesaf.services.sys.odb.tester;
```

```
public interface Line {
```

```
    String getName();
```

```
    void setName(String name);
```

```
    Location[] getPoints();
```

```
    Location getPoints(int index);
```

```
    void setPoints(Location[] points);
```

```
    void setPoints(int index, Location point);
```

```
    void setNextLine(Line nextLine);
```

```
    Line getNextLine();
```

```
}
```

- **Objects must contain attributes that are**
 - **Primitives (int, float, byte, double, boolean, short)**
 - E.g., name is a String
 - Includes instances of AbstractEnums
 - **Attributables (Encodable / Cloneable / Externalizable)**
 - **References to other ODB Objects**
 - E.g., nextLine is a reference to another Line object
 - References must be at the top level
 - **Arrays of any of the above**
 - E.g., points is an array of Locations