# C2SIM Server

Operation, Maintenance and Structure

Douglas Corner

George Mason University

C4I Cyber Center

dcorner@c4i.gmu.edu

+1 571-215-0773

# C2SIM Server

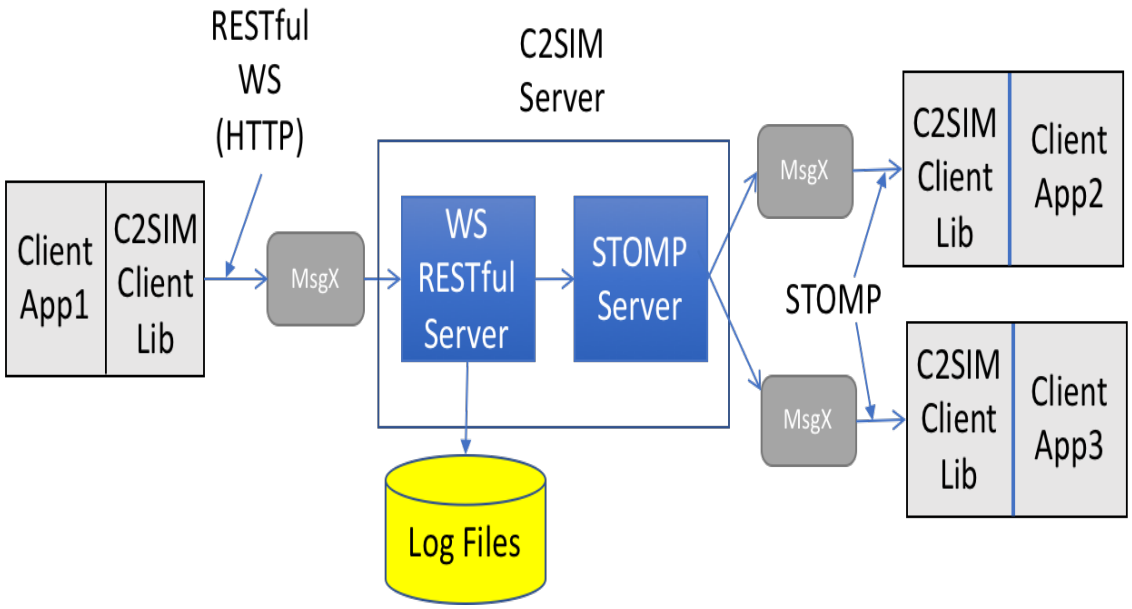
- The purpose of the C2SIM server is to distribute messages among processing elements of a Military C2-Simulation Coalition.
- Processing Elements
  - Command and Control Systems
  - Simulation Systems
  - System Operator(s)

# Overview

- Components
  - Server
    - RESTful WebServices server
      - Developed at GMU C4I-Cyber Center
      - All Java 8
      - 9,000 +lines of code
    - STOMP Publish Subscribe server
      - Off the shelf - Apache Apollo
  - Client Library
    - Required for all interfaces to server
    - Java - All environments
    - C++
  - Client Utilities
    - C2SIMGUI
    - Command line Utilities

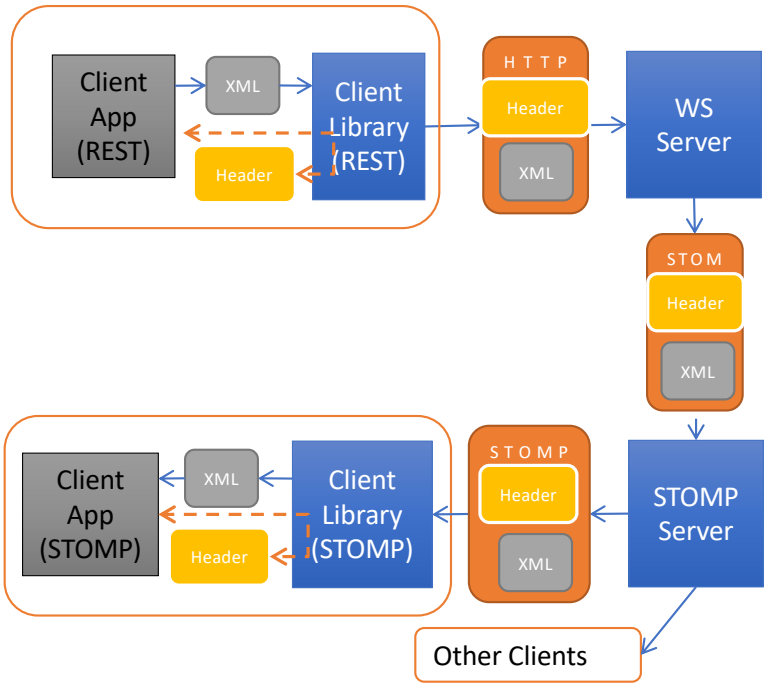
# Big Picture

NOTE: most Apps both send REST and receive STOMP



# Client Library Interaction

Generally a client will interact with both server components



# Operation

- Client Application creates message
- Client Application passes message to ClientLib
- Message submitted to Web Services server via REST
- REST Server processes message
- Server sends message to be published to STOMP server
- STOMP server sends message to all subscribed clients
- Client Library passes published message to client

# C2SIM Server Processing

- Message Types
  - Initialization Messages (C2SIM or MSDL)
  - Orders
  - Reports
  - Cyber Definitions – Defines attacks performed by server
- Commands
  - Submitted by control operator
  - Control server processing
  - Password protected

# C2SIM Server Components

- Primary Components
  - Centos 7 Linux Server running in a VMWare Virtual System
  - Tomcat 8.0.30 Application Server – accepts http input XML
  - Apache Apollo 1.7.1 – accepts subscriptions and distributes output XML
- Development Tools
  - Java 8.0.65
  - NetBeans 8.2 – Java IDE
  - Maven 3.3.9 – organizes components to simplify development
- Primary Server Components
  - JDOM 2.0.6 - parser
  - Apache log4j 2.**5 - logger**
  - Jersey  Servlet Container 2.5 – RESTful Web Services

# C2SIM Server Components

- System Layout
  - Centos 7
  - Server runs under user account bmluser
  - bmluser is in sudoers group – No need to use root account
  - Files under ~/c2simFiles
    - c2simCyber – Cyber Logs
    - c2simDebug – Debug Logs
    - c2simReplay – Replay Logs
    - C2SIMSchemaDB and C2SIMSchemaDB-Init used for translation mapping
    - schema – and schema/flattenedSchemas – Only used for translation mapping
    - Software – Software distribution files

# C2SIM Server Components

- File Layout
  - Applications are installed in /opt

  - /opt/tomcat/apache-tomcat-8.0.30 - Tomcat Root
  - /opt/tomcat/apache-tomcat-8.0.30/webapps – war file folder
    - C2SIMServer##4.x.y.z.war is primary Tomcat application
    - Small/secondary BML server for backward compatibility

  - /opt/bmlStomp/ - Apache Apollo Root (STOMP Messaging Server)

# C2SIM Client Library

- Essential to provide standard interface
- Isolates client users from changes in server interface
- Performs many detailed tedious functions
- Provides reasonable defaults
- Provided in both Java 8 and C++ (Windows) Versions

# C2SIM Library

- Classes
  - C2SIMClientREST_Lib
  - C2SIMClientSTOMP_Lib
  - C2SIMHeader  (FIPA Messaging)
  - C2SIMSTOMPMessage

# C2SIM Library Sample Code

- REST Interface

```
C2SIMClientREST_Lib c2s = new C2SIMClientREST_Lib(host, "ALL", "INFORM");
c2s.setHost(“localhost”);
c2s.setSubmitter(“dsc”);
C2s.setPort(“8080”);                    // Not necessary
try {
      String response = c2s.c2simRequest(“XML TEXT”);
}
catch(C2SIMClientException e) {
      System.out.println(“Error sending “ + e);
}
```

# C2SIM Library Sample Code

- STOMP Interface

```
C2SIMClientSTOMP_Lib c = new C2SIMClientSTOMP_Lib();
c.setHost(“localhost”);
c.setDestination("/topic/C2SIM");
try {
       C2SIMSTOMPMessage resp = c.connect();
       while (true)
          System.out.println(c.getNext_Block());
    }  // try
       catch (C2SIMClientException e) {
             System.out.println(“STOMP Error “ + e);
    }  // catch
```

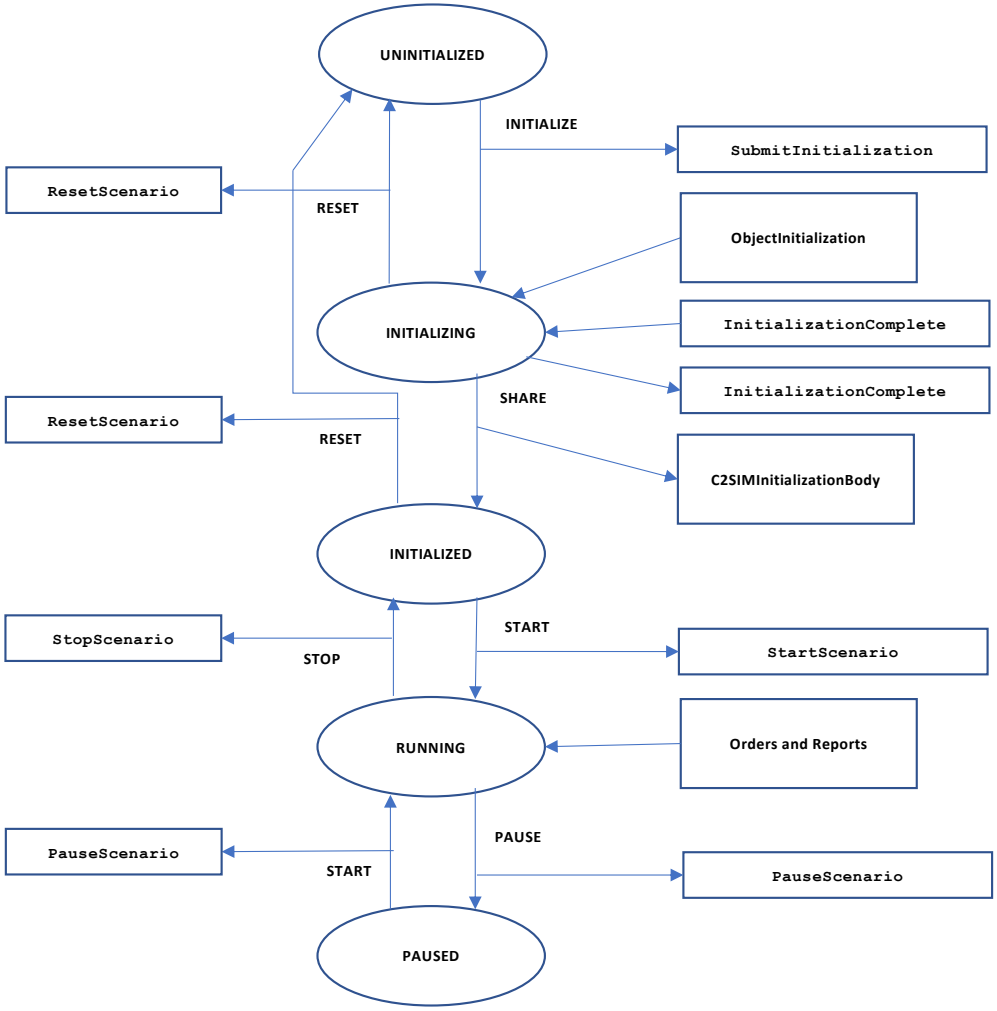
# C2SIM Client Command Line Utilities

- Users should use C2SIMGUI for production use
- Command Line Utilities helpful when debugging
- C2SIM_WSClient2 *(input to REST server, with responses)*

  java –jar C2SIM_WSClient2-4.7.0.2_ALL.jar localhost order3.xml dsc c2sim
- C2SIM_StompClient2 *(listen to STOM output stream)*

  java –jar C2SIM_StompClient2-4.7.0.1_ALL.jar localhost
- C2SIM_Command *(submit commends such as START STOP and SHARE)*
  - java -jar C2SIM_Command-4.7.0.0_ALL.jar localhost dsc START *password*
- C2SIM_Replay *(playback a logfile over STOMP)*
- B2BClient *(interconnect two servers – on each listen STOMP resend REST)*

# Server State

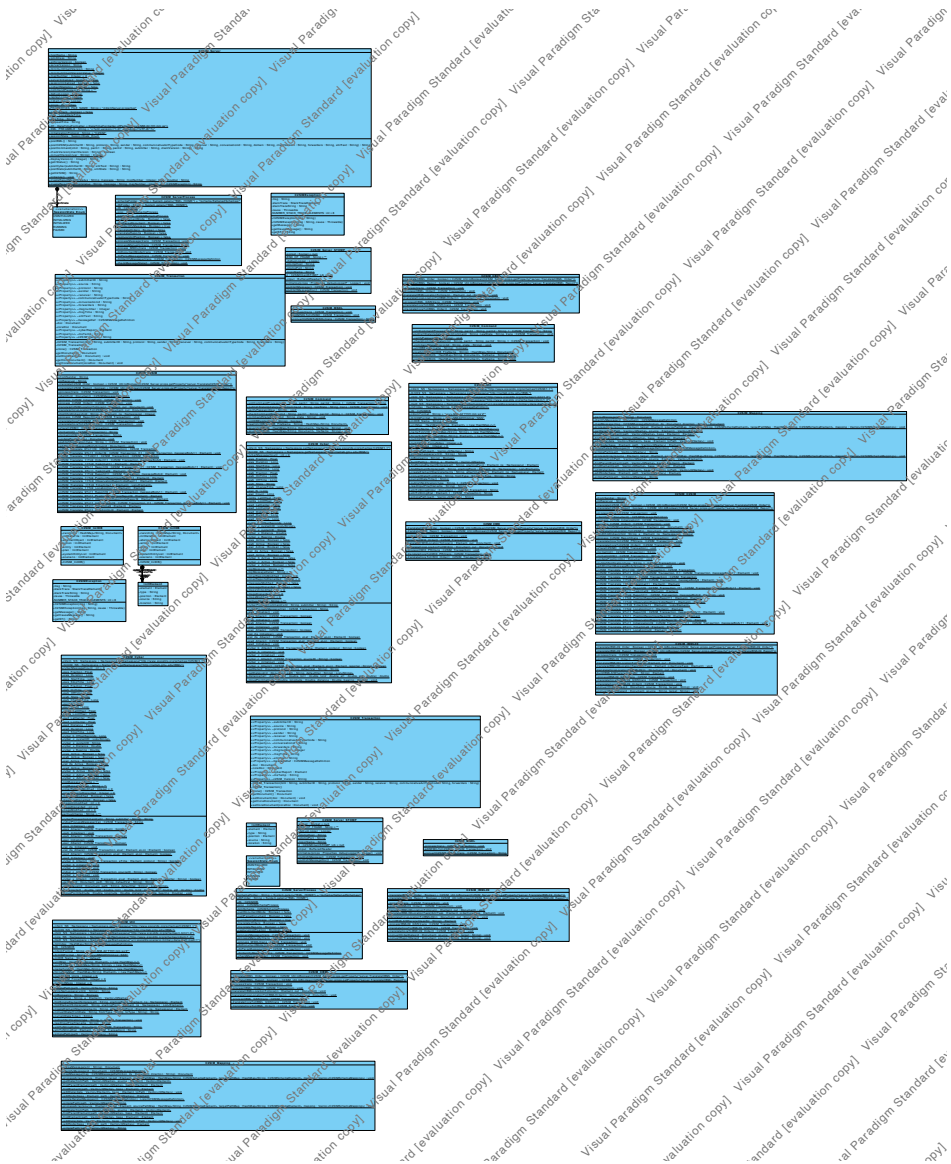


# DEMO

## C2SIM Server and Command Line Utilities

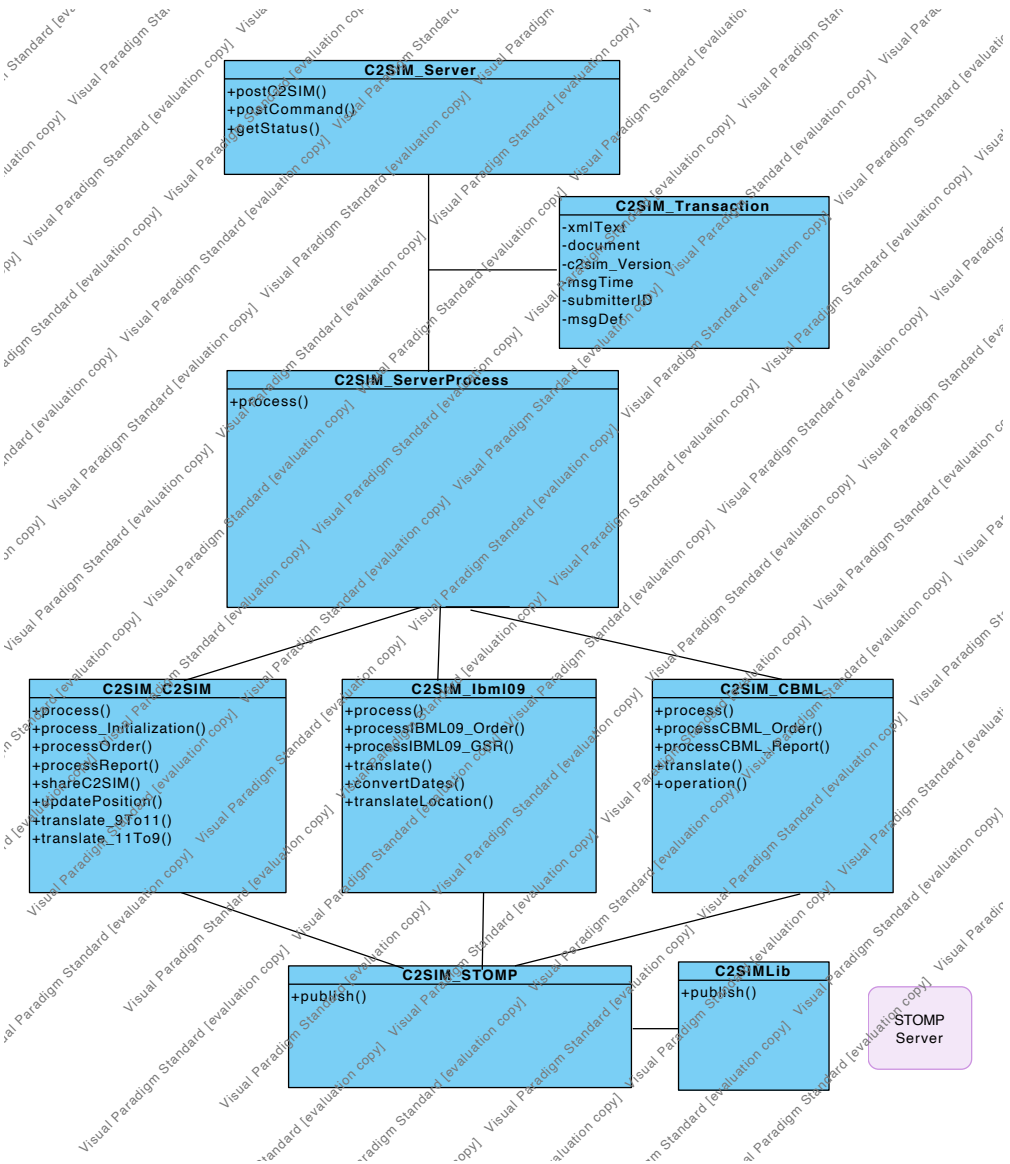
# UML Diagram

## All Classes



# UML Diagram

## Important Classes



# Code Review – C2SIM Server Web Services

# Server Operation

- Web Services Operation

  - URL is http://hostname:8080/C2SIMServer/c2sim
    - Submit C2SIM (IBML09, CBML) XML

  - Other URLs

    http://hostname:8080/C2SIMServer/status
    - Submit C2SIM (IBML09, CBML) XML

    http://hostname:8080/C2SIMServer/command
    - Submit server command

    http://hostname:8080/C2SIMServer/cyber
    - Submit Cyber Script

    http://hostname:8080/C2SIMServer/stats
    - Return response time from last c2sim post

# Server Flow

## Server follows these steps to process a message

| Class | Method | Function |
|---|---|---|
| | postC2SIM | Receive XML MSG and parameters from client |
| | | Create C2SIMTransaction |
| | | checkVersion |
| | | Check if MSG has already been handled (Multiserver Environment) |
| | | Log MSG to Replay Log |
| C2SIMServerProcess | processMessage | Check processing options |
| | processManually | Remove header from C2SIM/CWIX Messages |
| | | Identify message type |
| | | Log MSG and Type to Debug Log |
| | | Pass message through Cyber. Drop if directed |
| | checkMessageState | Check if this type message permitted in this state |
| | processManually | Call specific Class.process() depending on message type if translation is to be performed:<br>C2SIM_C2SIM.process<br>C2SIM_IBML09.process<br>C2SIM_CBML.process<br>C2SIM_CWIX.process<br>C2SIM_MSDL.process |
| | | Log message and identification to debugLogger |

# Demo – C2SIM Message Structure

# Message Flow

- Receive message
  - Create C2SIMTransaction
  - checkVersion() – Check ClientLib version
  - Log message exactly as received to ReplayLog
  - Remove C2SIM Header
  - Identify Message type and dialect),
    - Dialects – C2SIM (V0.0.9 and V1.00), IBML09, CBML-Light, MSDL
  - Check message type - See server state diagram

# Message Flow

- Initialization Processing - C2SIM
  - C2SIM V1.0 – Extract elements,  Add to Stored Initialization Data
  - C2SIM V0.0.9 – Translate to 1.0, Add to Stored Initialization Data
  - MSDL – Translate to 1.0, Add to Stored Initialization Data

- Initialization Processing – Share Command
  - Build C2SIMInitialization message from Stored Initialization Data and publish
  - Convert Stored Initialization Data to V0.0.9 and publish
  - Convert Stored Initialization Data to MSDL and publish

# Message Flow

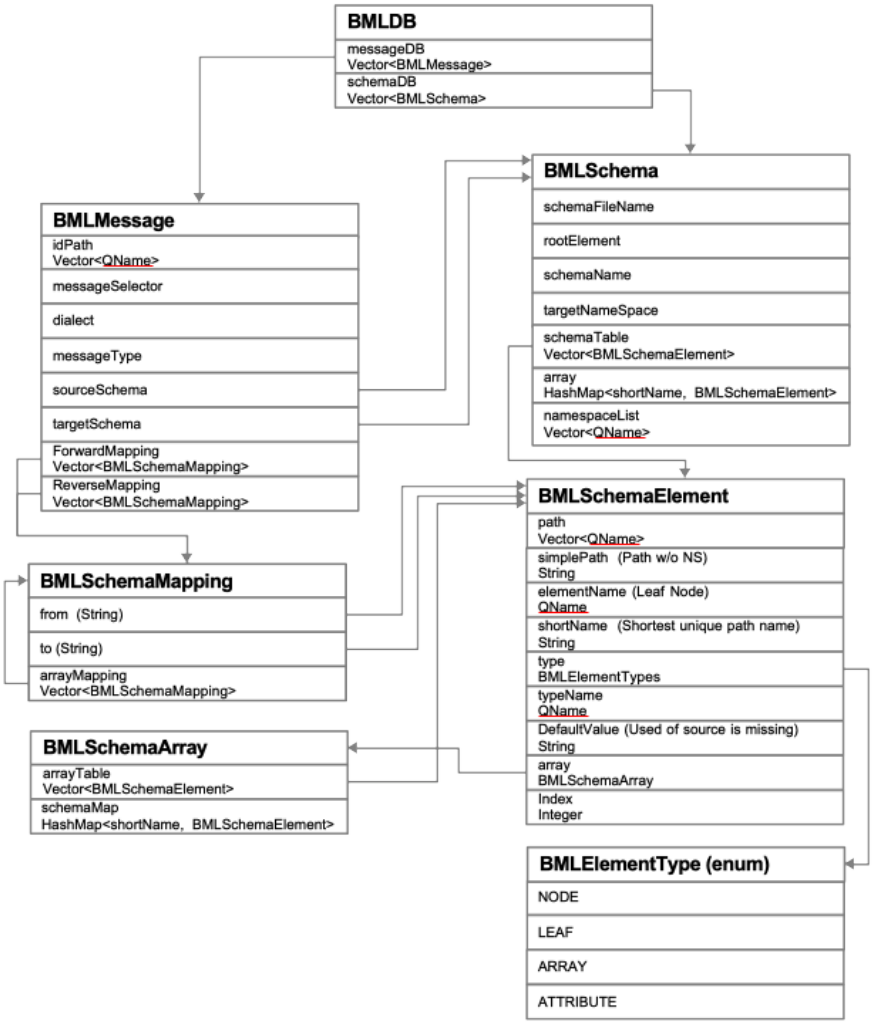
- Order Processing
  - Publish Order exactly as received
  - Translate Order to IBML09, CBML-Light, C2SIM-V0.0.9 as appropriate and publish
- Report Processing
  - Publish Report exactly as received
  - Translate Report to IBML09, CBML-Light, C2SIM V 0.0.9 as appropriate and publish
  - Record geographic position as current position

# Code Review – Server Processing

# Protocol Translation

- Initial Capability used field to field mapping
  - C2SIMSchema Class
    - Set of tables defining schema contents and mappings
  - C2SIMSchemaParser
    - Parses schema files (.xsd) produces tables
    - Uses Apache Schema API
  - C2SIMDB_Loader
    - Uses output of C2SIMSchemaParser
    - Creates tables in C2SIMSchema class defining mapping
  - C2SIM_Mapping Class in Server performs mapping
  - Most recent translations have been done with Java code
    - Mapping tables not robust enough
    - Difficulty with Apache Schema API and with C2SIMSchemaParser

# BMLServer 4 Table Structure




DSC 3/21/2017

# Code Review – Translation

# Server Configuration

- All configuration files are embedded in Web Archive - C2SIMServer.war
- Logging Configuration – log4j2.xml
- Server properties – c2simServer.properties
- Translation mappings - C2SIMSchemaDB
- Maven Project File – pom.xml
- When updating the server only one file (C2SIM_Server.war) is updated

# Server Properties File

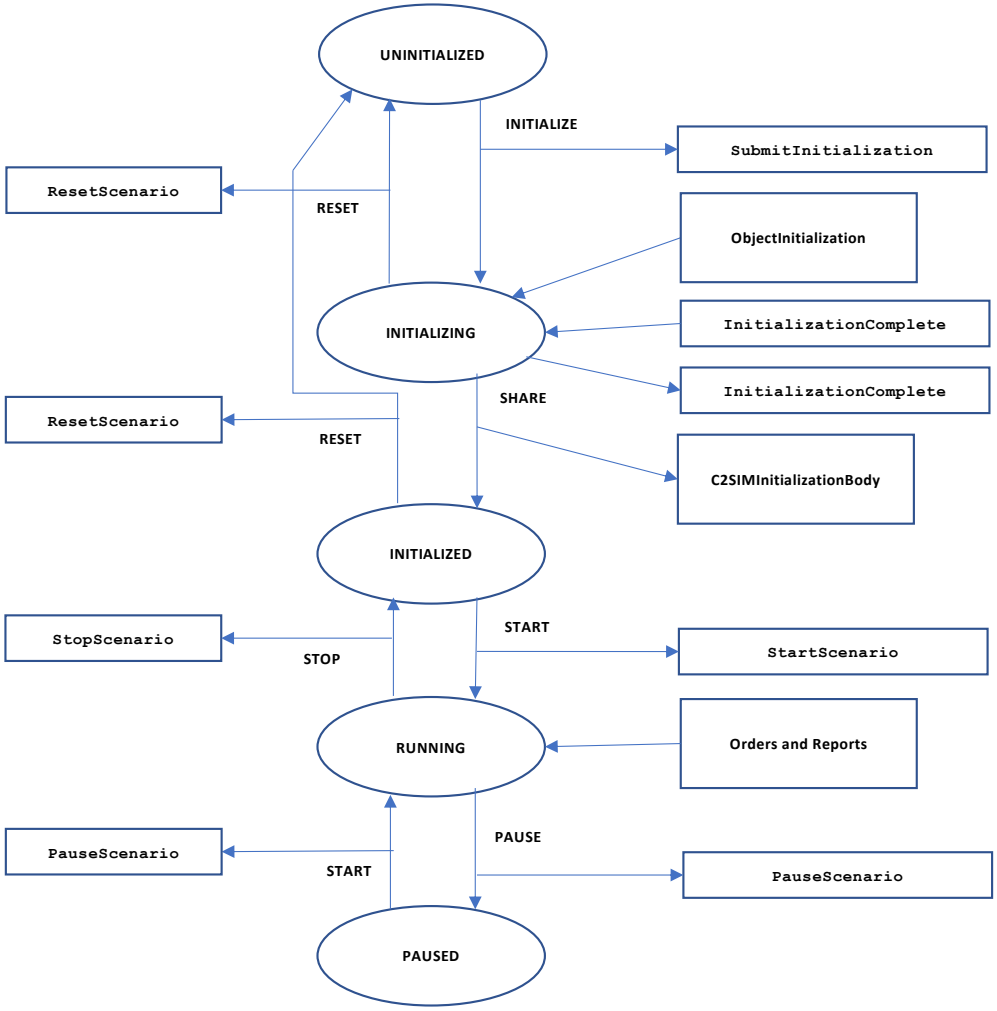
- stomp.serverHost=localhost
- stomp.port=61613
- stomp.topicName=/topic/C2SIM

- #Location of bmlFiles ($BML_HOME)
- server.bmlFiles = /home/bmluser/c2simFiles

- **DEMO – LOOK AT ACTUAL PROPERTIES FILE**

# Code Review – Initialization

# Server State



# Command Processing

- INITIALIZE
  - Send “SubmitInitialization” message
  - Set state to INITIALIZING
- SHARE
  - Send “InitializationComplete” message
  - Publish C2SIMInitializationBody messages
  - Set state to INITIALIZED
- START
  - Send “StartScenario” message
  - Set state to RUNNING

# Command Processing

- PAUSE
  - Send “PauseScenario” message
  - Set state to PAUSED
- STOP
  - Send “StopScenario” message
  - Set state to INITIALIZED
- RESET
  - Send “ResetScenario” message
- QueryInit
  - Resend Initialization data with latest coordinates

# Server Performance

- Tests run in June 2019
  - Multiple copies of client on host MAC
  - Server in VM on same system
  - Position reports used for test – By far the most common message
  - Throughput - 436 Messages per second
  - Certainly fast enough for most experiments
- Performance Improvements
  - Faster system
  - Move STOMP server to separate system

# Pending Work

Expected to be completed by May 27

- Completion of C2SIM V 0.0.9 translation
- Compliance with C2SIM 1.0.0
- Completion of JavaDOC for Server
- Final review of Server Operation Document
- Fourth C2SIM format in C2SIMGUI ServerValidation
  - 1.0.0, 0.0.9, CBML, IBML