



STO TECHNICAL REPORT

TR-MSG-136-Part-VII

Modelling and Simulation as a Service Volume 4: Experimentation Report

(Modélisation et Simulation en tant que service,
Volume 4 : rapport d'expérimentation.)

Developed by NATO MSG-136.



Published July 2019





STO TECHNICAL REPORT

TR-MSG-136-Part-VII

Modelling and Simulation as a Service Volume 4: Experimentation Report

(Modélisation et Simulation en tant que service,
Volume 4 : rapport d'expérimentation.)

Developed by NATO MSG-136.

The NATO Science and Technology Organization

Science & Technology (S&T) in the NATO context is defined as the selective and rigorous generation and application of state-of-the-art, validated knowledge for defence and security purposes. S&T activities embrace scientific research, technology development, transition, application and field-testing, experimentation and a range of related scientific activities that include systems engineering, operational research and analysis, synthesis, integration and validation of knowledge derived through the scientific method.

In NATO, S&T is addressed using different business models, namely a collaborative business model where NATO provides a forum where NATO Nations and partner Nations elect to use their national resources to define, conduct and promote cooperative research and information exchange, and secondly an in-house delivery business model where S&T activities are conducted in a NATO dedicated executive body, having its own personnel, capabilities and infrastructure.

The mission of the NATO Science & Technology Organization (STO) is to help position the Nations' and NATO's S&T investments as a strategic enabler of the knowledge and technology advantage for the defence and security posture of NATO Nations and partner Nations, by conducting and promoting S&T activities that augment and leverage the capabilities and programmes of the Alliance, of the NATO Nations and the partner Nations, in support of NATO's objectives, and contributing to NATO's ability to enable and influence security and defence related capability development and threat mitigation in NATO Nations and partner Nations, in accordance with NATO policies.

The total spectrum of this collaborative effort is addressed by six Technical Panels who manage a wide range of scientific research activities, a Group specialising in modelling and simulation, plus a Committee dedicated to supporting the information management needs of the organization.

- AVT Applied Vehicle Technology Panel
- HFM Human Factors and Medicine Panel
- IST Information Systems Technology Panel
- NMSG NATO Modelling and Simulation Group
- SAS System Analysis and Studies Panel
- SCI Systems Concepts and Integration Panel
- SET Sensors and Electronics Technology Panel

These Panels and Group are the power-house of the collaborative model and are made up of national representatives as well as recognised world-class scientists, engineers and information specialists. In addition to providing critical technical oversight, they also provide a communication link to military users and other NATO bodies.

The scientific and technological work is carried out by Technical Teams, created under one or more of these eight bodies, for specific research activities which have a defined duration. These research activities can take a variety of forms, including Task Groups, Workshops, Symposia, Specialists' Meetings, Lecture Series and Technical Courses.

The content of this publication has been reproduced directly from material supplied by STO or the authors.

Published July 2019

Copyright © STO/NATO 2019
All Rights Reserved

ISBN 978-92-837-2160-4

Single copies of this publication or of a part of it may be made for individual use only by those organisations or individuals in NATO Nations defined by the limitation notice printed on the front cover. The approval of the STO Information Management Systems Branch is required for more than one copy to be made or an extract included in another publication. Requests to do so should be sent to the address on the back cover.

Table of Contents

	Page
List of Figures	xiii
List of Tables	xv
List of Acronyms	xvi
MSG-136 Membership List	xvii
 Executive Summary and Synthèse	 ES-1
 Chapter 1 – Introduction	 1-1
1.1 Background	1-1
1.2 Objective	1-1
1.3 Document Overview	1-2
 Chapter 2 – Approach	 2-1
2.1 Experiments to Test the Reference Architecture	2-1
2.2 Experiments to Test Solutions of Simulation Services	2-2
2.3 Schema of Test Case Description	2-2
 Chapter 3 – Experiment on Containerized HLA Based Simulation Environment	 3-1
3.1 Overview	3-1
3.2 Test Case 1: Container Networking (NLD)	3-1
3.2.1 Objective/Topic/Question	3-1
3.2.2 Assumptions/Preconditions/Boundary Conditions	3-1
3.2.3 Systems and Interfaces	3-1
3.2.4 Test Setup	3-2
3.2.5 Processes and Activities	3-2
3.2.6 Observations	3-3
3.2.7 Outcome/Analysis	3-4
3.3 Test Case 2: Containerization of HLA Federates (AUS, NLD)	3-4
3.3.1 Objective/Topic/Question	3-4
3.3.2 Assumptions/Preconditions/Boundary Conditions	3-5
3.3.3 Systems and Interfaces	3-5
3.3.4 Test Setup	3-5
3.3.5 Processes and Activities	3-5
3.3.6 Observations	3-5
3.3.7 Outcome/Analysis	3-6
3.4 Test Case 3: Metadata Repositories and Discovery (GBR, DEU)	3-6
3.4.1 Objective/Topic/Question	3-6
3.4.1.1 Scope	3-6

3.4.2	Assumptions/Preconditions/Boundary Conditions	3-7
3.4.3	Systems and Interfaces	3-7
3.4.4	Experimental Setup	3-7
3.4.5	Processes and Activities	3-8
3.4.6	Observations	3-9
3.4.7	Outcome/Analysis	3-9
3.5	Test Case 4: Simulation Composition and Deployment (GBR)	3-9
3.5.1	Objective/Topic/Question	3-9
3.5.1.2	Scope	3-9
3.5.2	Assumptions/Preconditions/Boundary Conditions	3-9
3.5.3	Systems and Interfaces	3-9
3.5.4	Test Setup	3-10
3.5.4.1	Definitions	3-10
3.5.4.2	Composition	3-11
3.5.4.3	Deployment	3-11
3.5.4.4	Composition Concepts	3-11
3.5.5	Processes and Activities	3-15
3.5.5.1	Test #1	3-15
3.5.5.2	Test #2	3-21
3.5.5.3	Test #3	3-26
3.5.5.4	Test #4	3-28
3.6	Test Case 5: Container Orchestration Environments (NLD)	3-30
3.6.1	Objective/Topic/Question	3-30
3.6.2	Assumptions/Preconditions/Boundary Conditions	3-31
3.6.3	Systems and Interfaces	3-31
3.6.4	Test Setup	3-31
3.6.5	Processes and Activities	3-33
3.6.6	Observations	3-33
3.6.7	Outcome/Analysis	3-35
Chapter 4 – Simulation Services Experiments		4-1
4.1	Overview	4-1
4.2	Test Case 1: RPS, SES, and SimSys (DEU, NOR)	4-1
4.2.1	Objective/Topic/Question	4-1
4.2.2	Assumptions/Preconditions/Boundary Conditions	4-1
4.2.3	Used Systems and Interfaces	4-1
4.2.4	Test Setup	4-1
4.2.5	Processes and Activities	4-2
4.2.6	Observations	4-2
4.2.7	Outcome/Analysis	4-2
4.3	Test Case 2: Individual and Team Training for Naval C4ISTAR Operation (NATO-MSCOE)	4-3
4.3.1	Objective/Topic/Question	4-3
4.3.2	Assumptions/Preconditions/Boundary Conditions	4-3
4.3.3	Systems and Interfaces	4-3
4.3.4	Test Setup	4-3

4.3.5	Processes and Activities	4-4
4.3.6	Observations	4-4
4.3.7	Outcome/Analysis	4-5
4.4	Test Case 3: Individual Training of Radio/C2 Operator (NATO-MSCOE)	4-5
4.4.1	Objective/Topic/Question	4-5
4.4.2	Assumptions/Preconditions/Boundary Conditions	4-5
4.4.3	Systems and Interfaces	4-5
4.4.4	Test Setup	4-5
4.4.5	Processes and Activities	4-6
4.4.6	Observations	4-7
4.4.7	Outcome/Analysis	4-7

Chapter 5 – Summary and Conclusions **5-1**

Chapter 6 – References **6-1**

Annex A – Simulation Environment Agreements **A-1**

A.1	Metadata	A-1
A.1.1	Identification	A-1
A.2	Design	A-1
A.2.1	Scenario	A-1
A.2.1.1	Exercise Background	A-1
A.2.1.2	Situation	A-2
A.2.1.3	Phase 1: Small Craft Threat	A-3
A.2.1.4	Phase 2: Threat Intelligence Report	A-3
A.2.1.5	Phase 3: Course of Action Analysis	A-3
A.2.1.6	Phase 4: Direct Action Against Threats	A-6
A.2.1.7	Phase 5: SOF Secure Target Area	A-6
A.2.2	Conceptual Model	A-7
A.2.3	Architecture	A-16
A.2.4	Services	A-16
A.2.4.1	Weapon Service	A-16
A.2.4.2	Sensor Service	A-16
A.2.4.3	Damage Service	A-16
A.2.4.4	RWO Generation Service	A-18
A.2.5	Member Applications	A-19
A.3	Execution	A-19
A.3.1	Execution States	A-19
A.3.2	Time Management	A-19
A.3.3	Join and Resign	A-19
A.3.4	Update Rates	A-19
A.3.5	Performance Thresholds	A-19
A.3.6	Data Logging	A-19
A.3.7	Data Replay	A-19
A.3.8	Monitoring	A-20
A.3.9	Middleware Agreements	A-20
A.3.10	Member Configuration	A-20

A.4	Management	A-20
A.5	Data	A-20
A.5.1	Encodings	A-20
A.5.2	Data Exchange Models	A-20
A.5.3	Naming Conventions	A-20
A.5.4	Publish/Subscribe Responsibilities	A-20
A.6	Infrastructure	A-21
A.7	Modeling	A-21
A.8	Variances	A-21

Annex B – Docker Container Image Descriptions B-1

B.1	General	B-1
B.1.1	Home	B-1
B.1.1.1	MSaaS Docker Registry	B-1
B.1.1.2	Service Descriptions	B-1
B.1.2	MSaaS Docker Registry (TNO)	B-2
B.1.2.1	Obtain Access to the MSaaS Docker Registry	B-2
B.1.2.2	Request a User Account	B-2
B.1.2.3	Login to the MSaaS Docker Registry via the Web UI and Change Your Initial Password	B-2
B.1.2.4	Login to MSaaS Docker Registry via the Docker Command Line Interface	B-2
B.1.2.5	Push and Pull Images	B-2
B.1.2.6	Delete Images	B-3
B.2	Docker Image Descriptions	B-3
B.2.1	Cesium Image (TNO)	B-3
B.2.1.1	Image	B-3
B.2.1.2	Description	B-3
B.2.1.3	Synopsis	B-3
B.2.1.4	Docker Options	B-3
B.2.1.5	Container Options	B-3
B.2.1.6	Web Address	B-3
B.2.1.7	Example	B-3
B.2.2	Damage Server Image (TNO)	B-4
B.2.2.1	Image	B-4
B.2.2.2	Description	B-4
B.2.2.3	Synopsis	B-4
B.2.2.4	Docker Options	B-4
B.2.2.5	Container Options	B-5
B.2.2.6	Other Information	B-5
B.2.2.7	Example	B-6
B.2.3	EPIC Enhanced Perception and Integrated Control Image (LM)	B-7
B.2.3.1	Image	B-7
B.2.3.2	Description	B-7
B.2.3.3	Synopsis	B-8
B.2.3.4	Docker Options	B-9

B.2.3.5	Container Options	B-9
B.2.3.6	Other Information	B-9
B.2.3.7	Example	B-9
B.2.4	FEAT Editor Image (TNO)	B-9
B.2.4.1	Image	B-9
B.2.4.2	Description	B-9
B.2.4.3	Synopsis	B-9
B.2.4.4	Docker Options	B-10
B.2.4.5	Container Options	B-10
B.2.4.6	Other Information	B-10
B.2.4.7	Example	B-10
B.2.5	Google Chrome Image (TNO)	B-10
B.2.5.1	Image	B-10
B.2.5.2	Description	B-10
B.2.5.3	Synopsis	B-11
B.2.5.4	Docker Options	B-11
B.2.5.5	Container Options	B-11
B.2.5.6	Other Information	B-11
B.2.5.7	Example	B-11
B.2.6	Google Earth Image (TNO)	B-11
B.2.6.1	Image	B-11
B.2.6.2	Description	B-12
B.2.6.3	Synopsis	B-12
B.2.6.4	Docker Options	B-12
B.2.6.5	Container Options	B-12
B.2.6.6	Other Information	B-12
B.2.6.7	Example	B-12
B.2.7	KML Server Image (TNO)	B-13
B.2.7.1	Image	B-13
B.2.7.2	Description	B-13
B.2.7.3	Synopsis	B-14
B.2.7.4	Docker Options	B-14
B.2.7.5	Container Options	B-14
B.2.7.6	Other Information	B-14
B.2.7.7	Example	B-15
B.2.8	Logger Service Image (IFAD)	B-17
B.2.8.1	Image	B-17
B.2.8.2	Description	B-18
B.2.8.3	Synopsis	B-18
B.2.8.4	Docker Options	B-18
B.2.8.5	Container Options	B-18
B.2.8.6	Service Interface	B-18
B.2.8.7	Examples	B-19
B.2.8.8	Other Information	B-20
B.2.9	LRC Base Image (TNO)	B-20
B.2.9.1	Image	B-20

B.2.9.2	Description	B-21
B.2.9.3	Other Information	B-26
B.2.9.4	Example	B-28
B.2.10	MSaaS Portal Image (TNO)	B-28
B.2.10.1	Image	B-28
B.2.10.2	Description	B-28
B.2.10.3	Synopsis	B-28
B.2.10.4	Docker Options	B-28
B.2.10.5	Container Options	B-28
B.2.10.6	Other Information	B-29
B.2.10.7	Example	B-29
B.2.11	Munition Server Image (AUS, DST Group)	B-30
B.2.11.1	Image	B-30
B.2.11.2	Description	B-30
B.2.11.3	Synopsis	B-30
B.2.11.4	Docker Options	B-30
B.2.11.5	Container Options	B-31
B.2.11.6	Other Information	B-31
B.2.11.7	Example	B-31
B.2.12	MÄK License Manager Image (AUS, DST Group)	B-31
B.2.12.1	Image	B-31
B.2.12.2	Description	B-31
B.2.12.3	Synopsis	B-31
B.2.12.4	Docker Options	B-31
B.2.12.5	Container Options	B-31
B.2.12.6	Other Information	B-31
B.2.12.7	Example	B-32
B.2.13	MÄK RTI Image Structure (AUS, DST Group)	B-32
B.2.13.1	Overview	B-32
B.2.13.2	mak-rti:4.4.2	B-32
B.2.13.3	ma-lm	B-32
B.2.13.4	mak-rti-base	B-32
B.2.13.5	mak-rti-base	B-33
B.2.13.6	mak-rtiexec	B-33
B.2.13.7	ma-lrc-base	B-33
B.2.13.8	tl;dr	B-34
B.2.14	MÄK rtiexec Image (AUS, DST Group)	B-34
B.2.14.1	Image	B-34
B.2.14.2	Description	B-34
B.2.14.3	Synopsis	B-35
B.2.14.4	Docker Options	B-35
B.2.14.5	Container Options	B-35
B.2.14.6	Other Information	B-35
B.2.14.7	Example	B-35
B.2.15	MÄK VR Forces Image (TNO)	B-36
B.2.15.1	Image	B-36

B.2.15.2	Description	B-36
B.2.15.3	Synopsis	B-36
B.2.15.4	Docker Options	B-36
B.2.15.5	Container Options	B-36
B.2.15.6	Other Information	B-37
B.2.15.7	Example	B-37
B.2.16	Pacer Image (TNO)	B-37
B.2.16.1	Image	B-37
B.2.16.2	Description	B-38
B.2.16.3	Synopsis	B-38
B.2.16.4	Docker Options	B-38
B.2.16.5	Container Options	B-38
B.2.16.6	Other Information	B-39
B.2.16.7	Example	B-39
B.2.17	PDA Image (IFAD)	B-40
B.2.17.1	Image	B-40
B.2.17.2	Description	B-40
B.2.17.3	Synopsis	B-40
B.2.17.4	Docker Options	B-40
B.2.17.5	Container Options	B-40
B.2.17.6	Other Information	B-40
B.2.17.7	Example	B-40
B.2.18	Pitch CRC Image (TNO)	B-41
B.2.18.1	Image	B-41
B.2.18.2	Description	B-41
B.2.18.3	Synopsis	B-41
B.2.18.4	Docker Options	B-41
B.2.18.5	Container Options	B-42
B.2.18.6	Other Information	B-42
B.2.18.7	Example	B-42
B.2.19	Pitch DIS Adapter Image (TNO)	B-43
B.2.19.1	Image	B-43
B.2.19.2	Description	B-43
B.2.19.3	Synopsis	B-43
B.2.19.4	Docker Options	B-43
B.2.19.5	Environment Variables	B-43
B.2.19.6	Container Options	B-43
B.2.19.7	Other Information	B-43
B.2.19.8	Example	B-43
B.2.20	Pitch Recorder Image (TNO)	B-44
B.2.20.1	Image	B-44
B.2.20.2	Description	B-44
B.2.20.3	Synopsis	B-44
B.2.20.4	Docker Options	B-44
B.2.20.5	Container Options	B-45
B.2.20.6	Other Information	B-45

B.2.20.7	Example	B-45
B.2.21	Pitch WebGUI Image (TNO)	B-49
B.2.21.1	Image	B-49
B.2.21.2	Description	B-50
B.2.21.3	Synopsis	B-50
B.2.21.4	Docker Options	B-50
B.2.21.5	Container Options	B-50
B.2.21.6	Webview Session Information	B-50
B.2.21.7	Example	B-50
B.2.22	Proxy Image (TNO)	B-50
B.2.22.1	Image	B-50
B.2.22.2	Description	B-50
B.2.22.3	Synopsis	B-51
B.2.22.4	Docker Options	B-51
B.2.22.5	Container Options	B-51
B.2.22.6	Other Information	B-51
B.2.22.7	Example	B-51
B.2.23	Sensor Server Image (TNO)	B-51
B.2.23.1	Image	B-51
B.2.23.2	Description	B-51
B.2.23.3	Synopsis	B-52
B.2.23.4	Docker Options	B-52
B.2.23.5	Sensor Properties File	B-52
B.2.23.6	Used FOM Modules	B-53
B.2.23.7	How to Run the Sensor Server	B-53
B.2.24	SES Gulf (Service) Image (CPA ReDev GmbH)	B-54
B.2.24.1	Image	B-54
B.2.24.2	Description	B-54
B.2.24.3	Synopsis	B-54
B.2.24.4	Docker Compose Options	B-54
B.2.24.5	Container Options	B-54
B.2.24.6	Other Information	B-54
B.2.24.7	Example	B-54
B.2.25	SES Meppen (Service) Image (CPA ReDev GmbH)	B-55
B.2.25.1	Image	B-55
B.2.25.2	Description	B-55
B.2.25.3	Synopsis	B-55
B.2.25.4	Docker Compose Options	B-55
B.2.25.5	Container Options	B-55
B.2.25.6	Other Information	B-55
B.2.25.7	Example	B-55
B.2.26	SESDatabase Gulf Image (CPA ReDev GmbH)	B-56
B.2.26.1	Image	B-56
B.2.26.2	Description	B-56
B.2.26.3	Synopsis	B-56
B.2.26.4	Docker Options	B-56

B.2.26.5	Container Options	B-56
B.2.26.6	Other Information	B-56
B.2.26.7	Example	B-56
B.2.27	SESDatabase Meppen Image (CPA ReDev GmbH)	B-56
B.2.27.1	Image	B-56
B.2.27.2	Description	B-56
B.2.27.3	Synopsis	B-56
B.2.27.4	Docker Options	B-57
B.2.27.5	Container Options	B-57
B.2.27.6	Other Information	B-57
B.2.27.7	Example	B-57
B.2.28	ShipSim Image (AUS, DST Group)	B-57
B.2.28.1	Image	B-57
B.2.28.2	Description	B-57
B.2.28.3	Synopsis	B-57
B.2.28.4	Docker Options	B-58
B.2.28.5	Container Options	B-58
B.2.28.6	HTTP Server	B-60
B.2.28.7	Example	B-60
B.2.29	ShipUI Image (AUS, DST Group)	B-61
B.2.29.1	Image	B-61
B.2.29.2	Description	B-61
B.2.29.3	Synopsis	B-61
B.2.29.4	Docker Options	B-61
B.2.29.5	Container Options	B-62
B.2.29.6	Other Information	B-62
B.2.29.7	Example	B-62
B.2.30	Start Image (TNO)	B-62
B.2.30.1	Image	B-62
B.2.30.2	Description	B-63
B.2.30.3	Synopsis	B-63
B.2.30.4	Docker Options	B-63
B.2.30.5	Container Options	B-63
B.2.30.6	Other Information	B-63
B.2.30.7	Example	B-64
B.2.31	Symbol Service Image (IFAD)	B-66
B.2.31.1	Image	B-66
B.2.31.2	Description	B-66
B.2.31.3	Synopsis	B-66
B.2.31.4	Docker Options	B-67
B.2.31.5	Container Options	B-67
B.2.31.6	Service Interface	B-67
B.2.31.7	Examples	B-67
B.2.32	Syslog Image (TNO)	B-68
B.2.32.1	Image	B-68
B.2.32.2	Description	B-68

	B.2.32.3	Synopsis	B-68
	B.2.32.4	Docker Options	B-68
	B.2.32.5	Container Options	B-68
	B.2.32.6	Other Information	B-68
	B.2.32.7	Example	B-68
B.2.33		X Server Image (TNO)	B-69
	B.2.33.1	Image	B-69
	B.2.33.2	Description	B-69
	B.2.33.3	Synopsis	B-69
	B.2.33.4	Docker Options	B-69
	B.2.33.5	Container Options	B-70
	B.2.33.6	Other Information	B-70
	B.2.33.7	Example	B-70
B.3		FOMs	B-70
	B.3.1	Sensor FOM	B-70
	B.3.1.1	Module Overview	B-70
	B.3.1.2	Object Classes Overview	B-71
	B.3.1.3	Identification	B-71
	B.3.1.4	Object Classes	B-72
	B.3.1.5	Data Types	B-75
	B.3.2	Damage FOM	B-79
	B.3.2.1	Module Overview	B-79
	B.3.2.2	Interaction Classes Overview	B-80
	B.3.2.3	Identification	B-80
	B.3.2.4	Interaction Classes	B-81
B.4		Compositions	B-82
	B.4.1	Composition A: Big Single Node	B-82

List of Figures

Figure		Page
Figure 2-1	Experiment Environment	2-1
Figure 3-1	Single Docker Host	3-2
Figure 3-2	Multi Docker Host	3-2
Figure 3-3	AWS Cloud	3-3
Figure 3-4	Hybrid	3-3
Figure 3-5	Containerization by ‘Extension’	3-4
Figure 3-6	Containerization by ‘Composition’	3-5
Figure 3-7	The Ultimate Goal of the Work Carried Out in This Experiment is the Exchange of Data Between the UK and German Registry-Repository Capabilities	3-7
Figure 3-8	Experimental Setup	3-10
Figure 3-9	Example Composition	3-12
Figure 3-10	Sub-Composition Example	3-12
Figure 3-11	Sub-Composition Re-Use	3-13
Figure 3-12	MSaaS Information Model	3-13
Figure 3-13	Discover, Evaluate and Use	3-14
Figure 3-14	Harvesting Tool	3-15
Figure 3-15	Create Composition Process	3-15
Figure 3-16	Experiment Composition	3-16
Figure 3-17	Prototype HTTP Client Web Tool	3-16
Figure 3-18	Search and Discovery using HTTP Client Tool	3-19
Figure 3-19	Evaluate Associations Using HTTP Client Tool	3-19
Figure 3-20	Automate Using HTTP Client Tool	3-20
Figure 3-21	Exercise Control Sub-Composition	3-20
Figure 3-22	Composer Tool	3-21
Figure 3-23	Composition WPS	3-22
Figure 3-24	MSaaS Composer	3-23
Figure 3-25	MSaaS Composer, with Graphical Illustration	3-23
Figure 3-26	Topology View	3-24
Figure 3-27	MSaaS Composer	3-25
Figure 3-28	Topology View	3-25
Figure 3-29	Experiment Setup	3-26
Figure 3-30	AIMS Deployment Tool	3-27
Figure 3-31	Infrastructure as Code Deployment	3-28

Figure 3-32	Concurrent Event Setup	3-29
Figure 3-33	View of Event 1 Simulation	3-30
Figure 3-34	Test Setup	3-31
Figure 3-35	Docker Native Setup	3-32
Figure 3-36	Kubernetes Setup	3-32
Figure 3-37	Docker Native X-Server Composition	3-33
Figure 3-38	Kubernetes X-Server Replication Controller Specification	3-34
Figure 3-39	Kubernetes X-Server Service Specification	3-34
Figure 3-40	Compose UI	3-35
Figure 3-41	ElasticKube UI	3-35
Figure 4-1	Experimental Setup	4-2
Figure 4-2	Naval C4ISTAR Session Setup	4-3
Figure 4-3	Naval C4ISTAR Operations	4-4
Figure 4-4	Radio/C2 Operations	4-5
Figure 4-5	Radio/C2 Morpheus Operations	4-6
Figure A-1	Exercise Background	A-1
Figure A-2	Exercise Area: Strait of Hormuz and Gulf of Oman	A-2
Figure A-3	Situation Overview	A-2
Figure A-4	Phase 1 Situation	A-4
Figure A-5	Phase 2 Situation	A-4
Figure A-6	Phase 3 Situation	A-5
Figure A-7	Phase 4 Situation	A-6
Figure A-8	Phase 5 Situation	A-7
Figure A-9	Weapon Service Interactions	A-17
Figure A-10	Sensor Service Interactions	A-17
Figure A-11	Damage Service Interactions	A-18
Figure A-12	RWO Service Interactions	A-18
Figure B-1	MSaaS Portal UI	B-29
Figure B-2	Sensor Module Overview	B-71
Figure B-3	Sensor Object Classes Overview	B-71
Figure B-4	Sensor Object Classes	B-72
Figure B-5	Damage Module Overview	B-80
Figure B-6	Damage Interaction Classes Overview	B-80
Figure B-7	Damage Interaction Classes	B-81

List of Tables

Table		Page
Table 3-1	Test Cases Relation to Architecture Building Blocks	3-1
Table 3-2	Summary of Combinations	3-4
Table 3-3	List of Definitions	3-10
Table A-1	Exercise Entity List	A-3
Table A-2	List of Events	A-8
Table B-1	Environment Variables	B-21
Table B-2	Environment Variables	B-23
Table B-3	Environment Variables	B-24
Table B-4	Environment Variables	B-24
Table B-5	Environment Variables	B-26
Table B-6	Environment Variables	B-34
Table B-7	Web API	B-61
Table B-8	Module Description	B-71
Table B-9	Sensor Module Identification	B-71
Table B-10	HLAobjectRoot.Track Attributes	B-72
Table B-11	HLAobjectRoot.Track.AbsoluteTrack Attributes	B-73
Table B-12	Simple Data Types	B-75
Table B-13	TrackQualityEnum Enumerations	B-75
Table B-14	TrackIdentificationEnum Enumerations	B-76
Table B-15	TrackStatusEnum Enumerations	B-77
Table B-16	Array Data Types	B-77
Table B-17	TrackClassificationStruct Data Types	B-78
Table B-18	TrackPositionStruct Data Types	B-78
Table B-19	TrackVelocityStruct Data Types	B-79
Table B-20	TrackEntityTypeStruct Data Types	B-79
Table B-21	Module Overview	B-80
Table B-22	Damage Module Identification	B-80
Table B-23	HLAinteractionRoot.DamageReport.EntityDamageReport Parameters	B-81

List of Acronyms

ABB	Architecture Building Block
C2	Command and Control
DIS	Distributed Interactive Simulation
DSEEP	Distributed Simulation Engineering and Execution Process
FEAT	Federation Engineering Agreements Template
FOM	Federation Object Model
HLA	High Level Architecture
M&S	Modelling and Simulation
MOM	Message Oriented Middleware
MSaaS	M&S as a Service
RA	Reference Architecture
RTI	Run Time Infrastructure
SOA	Service Oriented Architecture

MSG-136 Membership List

CO-CHAIRS

Dr. Robert SIEGFRIED
aditerna GmbH
GERMANY
Email: robert.siegfried@aditerna.de

Mr. Tom VAN DEN BERG
TNO Defence, Security and Safety
NETHERLANDS
Email: tom.vandenberg@tno.nl

MEMBERS

LtCdr Tevfik ALTINALEV
Turkish Navy
TURKEY
Email: taltinalev@hotmail.com

Mr. Gultekin ARABACI
NATO JFTC
POLAND
Email: gultekin.arabaci@jftc.nato.int

Mr. Anthony ARNAULT
ONERA
FRANCE
Email: anthony.arnault@onera.fr

Col Thierry BELLOEIL
NATO ACT
UNITED STATES
Email: thierry.belloeil@act.nato.int

Dr. Michael BERTSCHIK
DEU Bundeswehr
GERMANY
Email: MichaelBertschik@bundeswehr.org

LtCol Dr. Marco BIAGINI
NATO M&S Centre of Excellence
ITALY
Email: mscoe.cde01@smd.difesa.it

Mr. Maxwell BRITTON
Department of Defence
AUSTRALIA
Email: maxwell.britton1@defence.gov.au

Dr. Solveig BRUVOLL
Norwegian Defence Research Establishment
NORWAY
Email: solveig.bruvoll@ffi.no

Dr. Pilar CAAMANO SOBRINO
CMRE
ITALY
Email: Pilar.Caamano@cmre.nato.int

Prof. Dr. Erdal CAYIRCI
Research Center for STEAM
TURKEY
Email: erdal@dataunitor.com

Mr. Turgay CELIK
MILSOFT Software Technologies
TURKEY
Email: tcelik@milsoft.com.tr

LtCol Roberto CENSORI
NATO M&S CoE
ITALY
Email: mscoe.ms08@smd.difesa.it

Maj Fabio CORONA
NATO M&S Centre of Excellence
ITALY
Email: mscoe.cde04@smd.difesa.it

Dr. Anthony CRAMP
Department of Defence
AUSTRALIA
Email: anthony.cramp@dst.defence.gov.au

Mr. Raphael CUISINIER
ONERA
FRANCE
Email: raphael.cuisinier@onera.fr

Mr. Efthimios (Mike) DOUKLIAS
Space and Naval Warfare Systems Center Pacific
UNITED STATES
Email: mike.d.douklias@navy.mil

Ing Christian FAILLACE
LEONARDO S.p.a.
ITALY
Email: christian.faillace@leonardocompany.com

Dr. Keith FORD
Thales
UNITED KINGDOM
Email: keith.ford@uk.thalesgroup.com

LtCol Stefano GIACOMOZZI
General Defence Staff
ITALY
Email: mscoe.ds02@smd.difesa.it

Mr. Sabas GONZALEZ GODOY
NATO ACT
UNITED STATES
Email: Sabas.Gonzalez@act.nato.int

Ms. Amy GROM
Joint Staff J7
UNITED STATES
Email: amy.m.grom.civ@mail.mil

Mr. Yannick GUILLEMER
French MoD
FRANCE
Email: yannick.guillemmer@intradef.gouv.fr

Dr. Jo HANNAY
Norwegian Defence Research Establishment (FFI)
NORWAY
Email: jo.hannay@ffi.no

Mr. Andrew HOOPER
MOD
UNITED KINGDOM
Email: andy.hooper321@mod.uk

Mr. Willem (Wim) HUIKAMP
TNO Defence, Security and Safety
NETHERLANDS
Email: wim.huiskamp@tno.nl

Dr. Frank-T. JOHNSEN
Norwegian Defence Research Establishment (FFI)
NORWAY
Email: frank-trethan.johnsen@ffi.no

LtCol Jason JONES
NATO M&S CoE
ITALY
Email: mscoe.dr02@smd.difesa.it

Lt Angelo KAIJSER
Dutch Ministry of Defence
NETHERLANDS
Email: AJ.Kaijser@mindef.nl

Mr. Daniel KALLFASS
EADS Deutschland GmbH/CASSIDIAN
GERMANY
Email: daniel.kallfass@airbus.com

Col Robert KEWLEY
West Point
UNITED STATES
Email: Robert.Kewley@usma.edu

LtCol Gerard KONIJN
Dutch Ministry of Defence
NETHERLANDS
Email: gerard.konijn@gmail.com

Mr. Niels KRARUP-HANSEN
MoD DALO
DENMARK
Email: nkh@mil.dk

Mr. Vegard Berg KVERNELV
Norwegian Defence Research Establishment (FFI)
NORWAY
Email: vegard.kvernelv@ffi.no

Capt Peter LINDSKOG
Swedish Armed Forces
SWEDEN
Email: peter.j.lindskog@mil.se

Mr. Jonathan LLOYD
Defence Science and Technology Laboratory (Dstl)
UNITED KINGDOM
Email: jplloyd1@dstl.gov.uk

Mr. Jose-Maria LOPEZ RODRIGUEZ
Nextel Aerospace, Defence and Security (NADS)
SPAIN
Email: jmlopez@nads.es

Mr. Rene MADSEN
IFAD TS A/S
DENMARK
Email: Rene.Madsen@ifad.dk

Ms. Sylvie MARTEL
NCIA
NETHERLANDS
Email: Sylvie.Martel@ncia.nato.int

Mr. Gregg MARTIN
Joint Staff J7
UNITED STATES
Email: gregg.w.martin.civ@mail.mil

Mr. Jose Ramon MARTINEZ SALIO
Nextel Aerospace, Defence and Security (NADS)
SPAIN
Email: jrmartinez@nads.es

LtCdr Mehmet Gokhan METIN
Navy Research Centre
TURKEY
Email: m_gokhan_metin@yahoo.com

Mr. Aljosa MILJAVEC
MoD, Slovenian Armed Forces
SLOVENIA
Email: Aljosa.Miljavec@mors.si

Mr. Brian MILLER
U.S. Army
UNITED STATES
Email: brian.s.miller116.civ@mail.mil

Dr. Katherine MORSE
John Hopkins University/APL
UNITED STATES
Email: Katherine.Morse@jhupl.edu

LtCol Ales MYNARIK
NATO JCBRN Defence COE
CZECH REPUBLIC
Email: mynarika@jcbncoe.cz

Mr. Rick NEWELL
JFTC
POLAND
Email: rick.newell@jftc.nato.int

Mr. Jeppe NYLOKKE
IFAD TS A/S
DENMARK
Email: jeppe.nylokke@ifad.dk

Mr. Robbie PHILLIPS
Lockheed Martin Corporation
AUSTRALIA
Email: robbie.phillips@lmco.com

Mr. Marco PICOLLO
Finmeccanica
ITALY
Email: marco.picollo@finmeccanica.com

Dr. LtCol (Ret) Dalibor PROCHAZKA
University of Defence
CZECH REPUBLIC
Email: dalibor.prochazka@unob.cz

Mr. Tomasz ROGULA
NATO Joint Force Training Centre
POLAND
Email: tomasz.rogula@jftc.nato.int

Dr. Martin ROTHER
IABG mbH
GERMANY
Email: rother@iabg.de

Mr. Angel SAN JOSE MARTIN
NATO ACT
UNITED STATES
Email: Angel.SanJoseMartin@act.nato.int

Maj Alfio SCACCIANOCE
NATO M&S CoE
ITALY
Email: mscoe.cde05@smd.difesa.it

LtCol Wolfhard SCHMIDT
JFTC
POLAND
Email: wolfhard.schmidt@jftc.nato.int

Mr. Barry SIEGEL
SPAWAR Systems Center – Pacific
UNITED STATES
Email: Barry.Siegel@navy.mil

Mrs. Louise SIMPSON
Thales
UNITED KINGDOM
Email: louise.simpson@uk.thalesgroup.com

Mr. Neil SMITH
UK MoD Dstl
UNITED KINGDOM
Email: nsmith@dstl.gov.uk

Mr. Per-Philip SOLLIN
Pitch Technologies AB
SWEDEN
Email: per-philip.sollin@pitch.se

Dr. Ralf STÜBER
CPA ReDev mbH
GERMANY
Email: stueber@supportgis.de

Capt Colin TIMMONS
Department of National Defence
CANADA
Email: colin.timmons@forces.gc.ca

Maj Dennis VAN DEN ENDE
Ministry of Defence
NETHERLANDS
Email: d.vd.ende@mindef.nl

Mr. Martin Dalgaard VILLUMSEN
IFAD TS A/S
DENMARK
Email: Martin.Villumsen@ifad.dk

Mr. Brian WARDMAN
Dstl Portsdown West
UNITED KINGDOM
Email: bwardman@dstl.gov.uk

Mr. Andrzej WNUK
Joint Warfare Centre
NORWAY
Email: andrzej.wnuk@jwc.nato.int

ADDITIONAL CONTRIBUTORS

Mr. Andy BOWERS
US Joint Staff J7
UNITED STATES
Email: francis.bowers@gdit.com

Mr. Brent MORROW
US Military Academy
UNITED STATES
Email: Brent.Morrow@usma.edu

Mr. Cory SAYLES
Lockheed Martin
UNITED STATES
Email: Cory.d.sayles@lmco.com

Mr. Roy SCRUDDER
The University of Texas at Austin
UNITED STATES
Email: roy.scrudder@arlut.utexas.edu

Mr. Dennis WILDE
European IAD Centre
UNITED STATES
Email: dennis.wilde@us.af.mil

Modelling and Simulation as a Service – Volume 4: Experimentation Report (STO-TR-MSG-136-Part-VII)

Executive Summary

NATO and nations use simulation environments for various purposes, such as training, capability development, mission rehearsal and decision support in acquisition processes. Consequently, Modelling and Simulation (M&S) has become a critical capability for the alliance and its nations. M&S products are highly valuable resources and it is essential that M&S products, data and processes are conveniently accessible to a large number of users as often as possible. However, achieving interoperability between simulation systems and ensuring credibility of results currently requires large efforts with regards to time, personnel and budget.

Recent developments in cloud computing technology and service-oriented architectures offer opportunities to better utilize M&S capabilities in order to satisfy NATO critical needs. M&S as a Service (MSaaS) is a new concept that includes service orientation and the provision of M&S applications via the as-a-service model of cloud computing to enable more composable simulation environments that can be deployed and executed on-demand. The MSaaS paradigm supports stand-alone use as well as integration of multiple simulated and real systems into a unified cloud-based simulation environment whenever the need arises.

NATO MSG-136 (“Modelling and Simulation as a Service – Rapid deployment of interoperable and credible simulation environments”) investigated the new concept of MSaaS with the aim of providing the technical and organizational foundations to establish the Allied Framework for M&S as a Service within NATO and partner nations. The Allied Framework for M&S as a Service is the common approach of NATO and nations towards implementing MSaaS and is defined by the following documents:

- Operational Concept Document;
- Technical Reference Architecture (including service discovery, engineering process and experimentation documentation); and
- Governance Policies.

MSG-136 evaluated the MSaaS concept in various experiments. The experimentation results and initial operational applications demonstrate that MSaaS is capable of realizing the vision that M&S products, data and processes are conveniently accessible to a large number of users whenever and wherever needed. MSG-136 strongly recommends NATO and nations to advance and to promote the operational readiness of M&S as a Service, and to conduct required Science & Technology efforts to close current gaps.

This document describes the MSG-136 experimentation activities that evaluated the MSaaS concept in two experiments with overall eight test cases. The Reference Architecture approach of MSG-136 proved to be valid; the technology MSG-136 used was well manageable.

The experimentation results demonstrate that MSaaS is capable of realizing the vision that M&S products, data and processes are conveniently accessible to a large number of users whenever and wherever needed.

Modélisation et Simulation en tant que service –

Volume 4 : Rapport d'expérimentation

(STO-TR-MSG-136-Part-VII)

Synthèse

L'OTAN et les pays membres utilisent les environnements de simulation à différentes fins, telles que la formation, le développement capacitaire, l'entraînement opérationnel et l'aide à la décision dans les processus d'acquisition. Par conséquent, la modélisation et simulation (M&S) est devenue une capacité cruciale pour l'Alliance et ses pays membres. Les produits de M&S sont des ressources extrêmement précieuses ; il est essentiel que les produits, données et procédés de M&S soient facilement accessibles à un grand nombre d'utilisateurs aussi fréquemment que possible. Toutefois, l'interopérabilité entre les systèmes de simulation et la crédibilité des résultats ne sont pas encore acquises et nécessitent beaucoup de temps, de personnel et d'argent.

Les évolutions récentes du cloud informatique et des architectures orientées service offrent l'occasion de mieux utiliser les capacités de M&S afin de répondre aux besoins cruciaux de l'OTAN. La M&S en tant que service (MSaaS) est un nouveau concept qui inclut l'orientation service et la fourniture d'applications de M&S via le modèle « en tant que service » du cloud informatique, dans le but de proposer des environnements de simulation plus faciles à composer et pouvant être déployés et exécutés à la demande. Le paradigme du MSaaS permet aussi bien une utilisation autonome que l'intégration de multiples systèmes simulés et réels au sein d'un environnement de simulation dans le cloud, chaque fois que le besoin s'en fait sentir.

Le MSG-136 de l'OTAN (« Modélisation et simulation en tant que service (MSaaS) – Déploiement rapide d'environnements de simulation crédibles et interopérables ») a étudié le nouveau concept de MSaaS afin de fournir les bases techniques et organisationnelles permettant d'établir le « cadre allié de M&S en tant que service » au sein de l'OTAN et des pays partenaires. Le cadre allié de M&S en tant que service est la démarche commune de l'OTAN et des pays visant à mettre en œuvre la MSaaS. Il est défini dans les documents suivant :

- Document de définition opérationnelle ;
- Architecture de référence technique (incluant la communication du service, le processus d'ingénierie et la documentation d'expérimentation) ; et
- Politiques de gouvernance.

Le MSG-136 a évalué le concept de MSaaS au moyen de diverses expériences. Les résultats d'expérimentation et les premières applications opérationnelles démontrent que la MSaaS est capable de rendre les produits, données et processus de M&S commodément accessibles à un grand nombre d'utilisateurs, quels que soient l'endroit et le moment où le besoin s'en fait sentir. Le MSG-136 recommande vivement à l'OTAN et aux pays de faire progresser et d'améliorer l'état de préparation opérationnelle de la M&S en tant que service et de mener les travaux de science et technologie requis pour combler les lacunes actuelles.

Ce document décrit les activités d'expérimentation du MSG-136 qui ont évalué le concept de MSaaS lors de deux expériences couvrant au total huit cas d'essai. La démarche d'architecture de référence du MSG-136 a prouvé sa validité ; la technologie utilisée était très maniable.

Les résultats d'expérimentation démontrent que la MSaaS est capable de rendre les produits, données et processus de M&S commodément accessibles à un grand nombre d'utilisateurs, quels que soient l'endroit et le moment où le besoin s'en fait sentir.



Chapter 1 – INTRODUCTION

1.1 BACKGROUND

NATO and the nations use distributed simulation environments for various purposes, such as training, mission rehearsal, or decision support in acquisition processes. Achieving interoperability between participating simulation systems and ensuring credibility of results still requires enormous efforts with regards to time, personnel, and budget.

Recent technical developments in the area of cloud computing technology and Service Oriented Architectures (SOA) may offer opportunities to better utilize M&S capabilities to satisfy NATO critical needs. A new concept that includes service orientation and the provision of M&S applications *via* as-a-service cloud computing may enable more composable simulation environments that can also be deployed more rapidly and on-demand. This new concept is known as “M&S as a Service”. (MSaaS).

NATO MSG-136 (“Modelling and Simulation as a Service – Rapid deployment of interoperable and credible simulation environments”) investigates this new concept with the aim to provide the technical and organizational foundations for a future permanent service-based M&S Ecosystem within NATO and partner nations.

MSG-136 focuses on several areas of M&S as a Service within NATO:

- **Governance:** The governance concept and roadmap for M&S as a Service within NATO;
- **Operational:** The operational concept of M&S as a Service: how does it work from the user point of view; and
- **Technical:** The technical concept of M&S as a Service, covering reference architecture, reference services and reference engineering process.

This document, Volume 4, Experimentation, provides details regarding experiments conducted in support of the technical concept development. Other technical area documents developed are:

- Volume 1, Technical Reference Architecture;
- Volume 2, Service Discovery and Metadata; and
- Volume 3, Reference Engineering Process.

1.2 OBJECTIVE

This document presents the M&S as a Service (MSaaS) Experimentation Plan and Experimentation Results for the experiments conducted by MSG-136. Plan and results are combined in a single document to reduce the amount of documentation, and to keep related information closely together.

The objective of the experimentation is to *test* Architecture Building Blocks (ABBs) from the MSaaS Technical Reference Architecture (Volume 1) and associated approaches for Discovery Service and Metadata (Volume 2).

Key questions to be answered are:

- 1) What Solution Building Blocks can be identified for Architecture Building Blocks?
- 2) Do the Architecture Building Blocks sufficiently cover the Solution Building Blocks used in the experiments?

1.3 DOCUMENT OVERVIEW

In Chapter 2, we describe the general approach for doing tests and experiments. Chapter 3 is about the containerized HLA experiments, Chapter 4 about the simulation services experiments. In Chapter 5, we draw a summary and conclusions.

Chapter 2 – APPROACH

The experimentation plan defines two strands of experiments: experiments to test the Reference Architecture, and experiments to test solutions for the Simulation Services. Experiment test cases are defined and performed, and results recorded. A brief overview of the two strands of experiments is provided in the following sections.

2.1 EXPERIMENTS TO TEST THE REFERENCE ARCHITECTURE

This strand of experiments concerns the testing of the Reference Architecture, which is the enabling technology for ABBs in the Reference Architecture. In summary the experiment is characterized as follows:

- **Experiment:** “Containerized HLA based simulation environment”:
 - **Simulation Services Implementation:** Containerized HLA federates, containerized web services;
 - **Simulation Services Interfaces:** HLA-RTI, web service interfaces; and
 - **M&S Message-Oriented Middleware Services:** HLA-RTI.

The experiment environment for the experiment is illustrated in Figure 2-1. The experiment environment is a collection of private clouds and a common cloud. The common cloud¹ can be used by MSG-136 members for tests performed in each experiment.

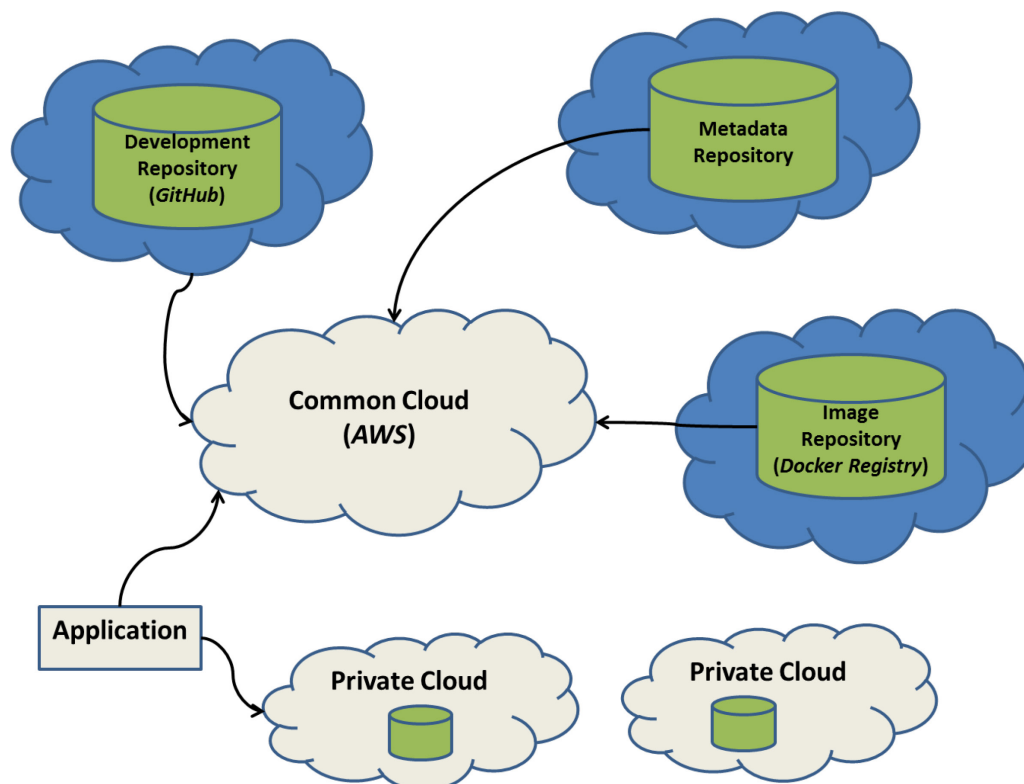


Figure 2-1: Experiment Environment.

¹ Sponsored by NATO CSO.

APPROACH

In addition, the following common components are provided in the experiment environment:

- Docker Registry and web-based front-end for the exchange of Docker container images (provided by NLD); and
- GitHub repository for the description of services in the Docker Registry, and for the exchange of software, configuration files and other developmental data (provided by USA).

Container technology is beneficial since it provides a process execution contract between solution providers and consumers. A software solution is provided as a container along with necessary dependencies with the solution consumer only required to provide a host system running the relevant container engine. This removes the need for the solution provider to account for all possible consumer environment or, alternatively, for the solution consumer to specifically tailor a host system for a specific provided solution. In the NATO context with multiple, diverse nations, having this common process execution environment greatly simplifies the infrastructure aspects of these experiments.

Docker is the container technology used throughout these experiments. While container concepts have existed for a few decades, it was the release of Docker in 2013 that has spurred its use. The rapid uptake of container technology has spawned a cross industry standards activity known as the Open Container Initiative (OCI). The fact that the OCI based its work from the Docker technology stack is testament to the popularity and usefulness of the Docker container implementation.

2.2 EXPERIMENTS TO TEST SOLUTIONS OF SIMULATION SERVICES

The second strand of experiments tests solutions for Simulation Services, one of the main ABBs in the Reference Architecture. These experiments focus on service interfaces and service interactions.

This strand of experiments includes tests with the following solutions for the Simulation Services:

- Synthetic Environment Service (i.e., GER SES);
- Route Planning Service (i.e., NOR RPS);
- Cloud Orchestration Solution (i.e., MSCOE – Leonardo ITA OCEAN); and
- Computer Generated Forces Service and C2-Sim gateway Service (i.e., ITA SGA and Gateway).

2.3 SCHEMA OF TEST CASE DESCRIPTION

The test cases and test results are described by using the following standard structure:

- **Objective/Topic/Question:** Objective of the test case, what parts of the Reference Architecture are addressed.
- **Assumptions/Preconditions/Boundary Conditions:** Assumptions etc. necessary to conduct the test.
- **Systems and Interfaces:** Systems/member applications that are involved in the test.
- **Test Setup:** Description of the test setup (e.g., picture of network or simulation environment).
- **Processes and Activities:** Events and activities conducted during the test.
- **Observations:** Any observations on system response, system behavior, planned and unplanned intervention by experimenter, etc.
- **Outcome/Analysis:** Has the objectives of the test case been achieved, and was the initial question answered? Any problems, difficulties, etc.

Chapter 3 – EXPERIMENT ON CONTAINERIZED HLA BASED SIMULATION ENVIRONMENT

3.1 OVERVIEW

The Experiment “Containerized HLA Based Simulation Environment” is broken down in several test cases, where each test case addresses specific parts of the Reference Architecture. An overview of test cases and related Architecture Building Blocks is provided in Table 3-1.

Table 3-1: Test Cases Relation to Architecture Building Blocks.

Test Case	Related Architecture Building Blocks
Test Case 1: Container Networking	<ul style="list-style-type: none"> • Communication Services (RA Layer 1) • M&S Message-Oriented Middleware Services
Test Case 2: Containerization of HLA Federates	<ul style="list-style-type: none"> • Modelling Services
Test Case 3: Metadata Repositories and Discovery	<ul style="list-style-type: none"> • M&S Repository Services • M&S Registry Services
Test Case 4: Simulation Composition	<ul style="list-style-type: none"> • M&S Composition Services
Test Case 5: Container Orchestration Environments	<ul style="list-style-type: none"> • Communication Services (RA Layer 1) • M&S Composition Services • M&S Repository Services • Security Services (RA Layer 1 and 2) • Service Management and Control Services (RA Layer 2)

3.2 TEST CASE 1: CONTAINER NETWORKING (NLD)

3.2.1 Objective/Topic/Question

Federates in an HLA federation typically rely on network connections to enable communication. Depending on the RTI, a mixture of connection and connection less and directed or broadcast connections may be used. Docker provides a number of networking options by default and the ability to write custom network plug-ins where required. Docker supports several different methods for networking containers: host, bridge, and overlay.

The objective of this test is to investigate the bridge and overlay networking models for connecting containerized HLA federate applications.

3.2.2 Assumptions/Preconditions/Boundary Conditions

None.

3.2.3 Systems and Interfaces

Simple HLA federate applications are used, since the objective is to merely investigate Docker network models.

3.2.4 Test Setup

The test environment consists of:

- Two Linux VMs with a Docker Engine, running on one host system; and
- One Amazon AWS Linux VM with a Docker Engine.

3.2.5 Processes and Activities

The following combinations have been setup and tested.

3.2.5.1 Single Docker Host

As shown in Figure 3-1:

- Simulation environment composed of containers running in a single Docker Host.
- Network: Default docker bridge.
- HLA-RTI: Pitch (TCP), Portico (Multicast).

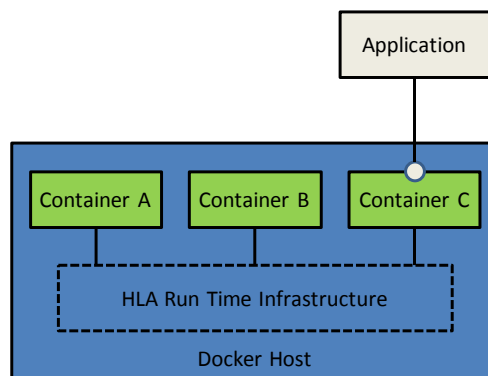


Figure 3-1: Single Docker Host.

3.2.5.2 Multi Docker Host

As shown in Figure 3-2:

- Simulation environment composed of containers running in a cluster of Docker Hosts in a LAN.
- Network: Default docker overlay.
- HLA-RTI: Pitch (TCP), Portico (Multicast).

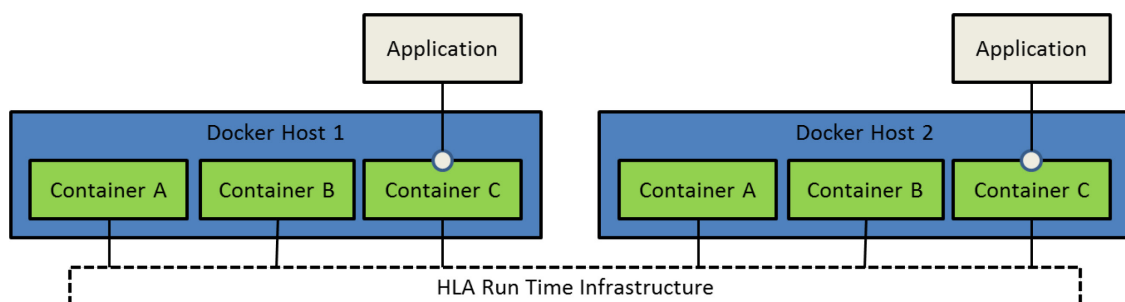


Figure 3-2: Multi Docker Host.

3.2.5.3 AWS Cloud

As shown in Figure 3-3:

- Simulation environment composed of containers running in a cluster of Docker Hosts both in the Amazon Cloud and in a LAN.
- Network: Weave overlay network (<http://weave.works>).
- HLA-RTI: Pitch (TCP), Portico (Multicast).

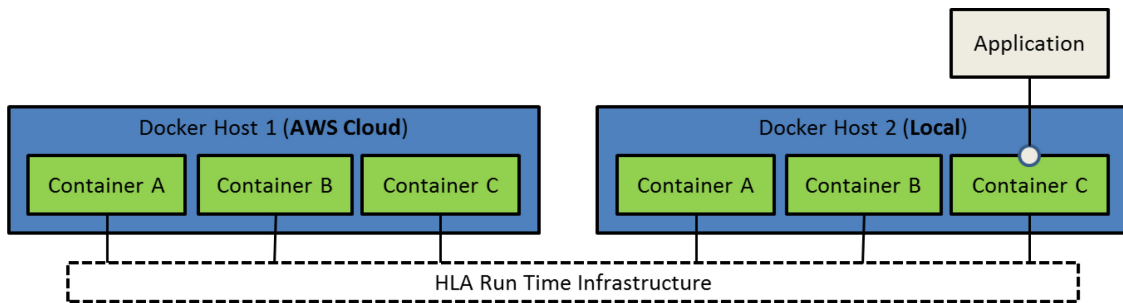


Figure 3-3: AWS Cloud.

3.2.5.4 Hybrid

As shown in Figure 3-4:

- Simulation environment composed of containerized federates running in a Docker Host and of non-containerized federates running on the host system, all in a LAN.
- Network: Weave overlay network (<http://weave.works>).
- HLA-RTI: Pitch (TCP), Portico (Multicast).

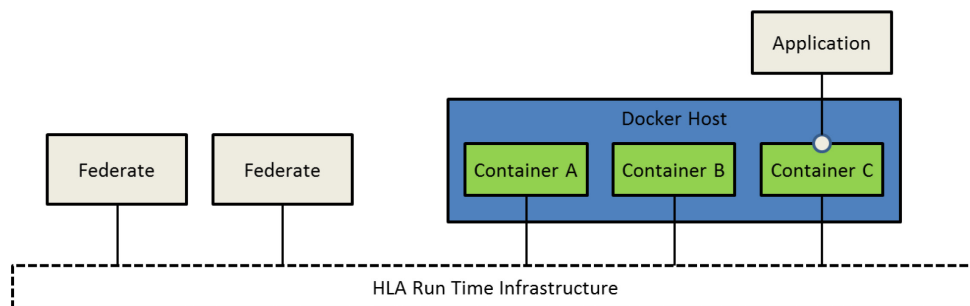


Figure 3-4: Hybrid.

3.2.6 Observations

Multicast is not supported by the default docker overlay network. Only Weave overlay supports multicast.

For the AWS Cloud case the Weave network setup has to be instantiated from the local system due to a corporate firewall.

Weave allows the network endpoint to be extended to the host (Linux) operating system, allowing non-containerized applications to connect to the overlay network.

By manually editing IP routes and IP tables it is possible to extend the number of combinations, e.g., mixing non-containerized Windows applications with containerized Linux applications.

3.2.7 Outcome/Analysis

A summary of successful combinations is provided in Table 3-2.

Table 3-2: Summary of Combinations.

Network Driver	Single Host	Multi Host	AWS Cloud	Hybrid
Docker bridge	TCP, Multicast	–	–	–
Docker overlay	–	TCP	TCP	–
Weave overlay	–	TCP, Multicast	TCP, Multicast	TCP, Multicast

3.3 TEST CASE 2: CONTAINERIZATION OF HLA FEDERATES (AUS, NLD)

3.3.1 Objective/Topic/Question

The objective of this test case is to evaluate approaches in containerizing HLA federate applications.

Two approaches – design patterns – to containerize an HLA federate application are:

- Containerization by ‘extension’ (as shown in Figure 3-5):
- Makes use of the Docker Image hierarchy to link a federate application with a base LRC.
 - One Image: build Federate Application Image from an LRC base image and add the application code in a new file system layer.
 - LRC combined with federate application at *image build time*.

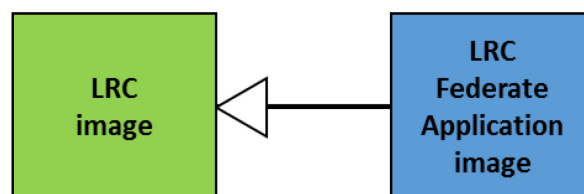


Figure 3-5: Containerization by ‘Extension’.

- Containerization by ‘composition’ (as shown in Figure 3-6):
 - Makes use of Docker Volumes to achieve the linkage.
 - Two Images: LRC Base Image and Federate Application Image.
 - Federate Application Image just contains the federate application; based on a small base image that provides rudimentary file system operations.
 - Installed directory is exposed as a volume that is mounted into the LRC container acting as a plugin manager for application code.
 - LRC combined with federate application at *image run time*.

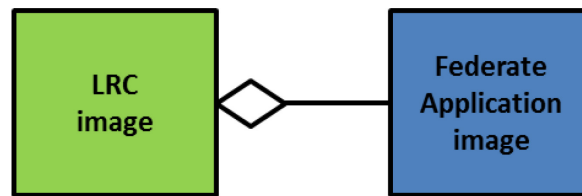


Figure 3-6: Containerization by 'Composition'.

The design patterns aim at modularity and reuse, especially when 'linking' with a Local RTI Component (LRC). Changing LRC should be seamless to the federate application.

These patterns are described in Ref. [1].

3.3.2 Assumptions/Preconditions/Boundary Conditions

LRC base containers for Pitch, MaK and Portico are available.

3.3.3 Systems and Interfaces

Several federate applications are used in the test case:

- MaK VR-Forces;
- Pitch Recorder; and
- A number of Java federate applications, including KML Server and Shimsim.

3.3.4 Test Setup

The test environment consists of a single Linux VM running a Docker Engine.

3.3.5 Processes and Activities

Activities for both approaches:

- Create Dockerfile;
- Build container image;
- Create Docker compose file; and
- Start simulation composition.

3.3.6 Observations

For all federate applications it was fairly easy to create a Dockerfile and build a Docker image.

A MaK VR-Forces container image was only built with the extension pattern, using the Pitch LRC base image.

The Pitch Recorder container image was only built with the extension pattern, using both the Pitch LRC base image and the Portico LRC base image. However, it was not possible to get the Portico based image to work with other federates (Recorder federate could create/join a federation execution, but did not discover other federates).

Container images for the other Java applications were built with both patterns, for all available LRC base images.

3.3.7 Outcome/Analysis

Both patterns are very useful, in particular the extension pattern. It takes away the burden to install an LRC and provides a standardized set of environment variables to interact with the LRC. The extension pattern is most easy to use and results in fewer containers in a simulation execution plus smaller Docker compose files.

Various useful features are present in the LRC base images, some are generic (independent of RTI type), others are dependent on RTI type, such as:

- Bootstrapping of federate applications (generic);
- Randomized start times of federate applications (generic); and
- Network adapter and log level settings.

3.4 TEST CASE 3: METADATA REPOSITORIES AND DISCOVERY (DEU, GBR)

3.4.1 Objective/Topic/Question

- Assess overall approach to discovery and identify complete approach for describing, defining and specifying M&S resources.
- Evaluate MSG-136 Service Description Template (SST) (in terms of structure, completeness, unambiguity, etc.) as a method of producing encoded simulation objects:
 - Review MSG-136 SST.
- Evaluate the ability to exchange metadata between Registries/Repositories from different nations, and to share M&S resources between MSaaS ecosystems:
 - Provide advice on approach to metadata, its role within a wider SOA ecosystem, and required metadata standards; and
 - Provide advice on how discovery enables resource sharing and simulation composition.
- Evaluate ability to Discover and obtain M&S Resources (Service/Data/Asset) using Registries/Repositories from different nations:
 - Identify limitations, issues;
 - Provide advice on required standards e.g., interfaces; and
 - Compare GBR/Aditerna approaches to Discovery.

3.4.1.1 Scope

- Evaluate MSaaS ecosystem elements relevant to discovery.
- MSaaS ABBs to be evaluated:
 - M&S Metadata Repository Services; and
 - M&S Discovery Services.
- Cross-cutting elements to be evaluated:
 - Information layer; and
 - Governance layer.

3.4.2 Assumptions/Preconditions/Boundary Conditions

- UK MSaaS GeoRegistry operating in cloud environment with external documented API. Accessible to external users who are cleared for use and allocated an account.
- Aditerna SRP Registry/Repository operating in cloud environment with external documented API. Accessible to external users who are cleared for use and allocated an account.
- An agreed set of simulation resources with associated encoded simulation objects and metadata.

3.4.3 Systems and Interfaces

- Systems:
 - UK GeoRegistry; and
 - Aditerna SRP Tool.
- Interfaces:
 - GeoRegistry API.
- Data:
 - Schema for MSG-136 Service Description Template;
 - Schema for AIMs M&S information objects and metadata; and
 - Metadata for national resources in the format defined by the generating nation.

3.4.4 Experimental Setup

The purpose of the experimentation is to demonstrate the feasibility of sharing information between different Registries (see Figure 3-7). However, as the German capability does not have an external API, two-way querying cannot be achieved.

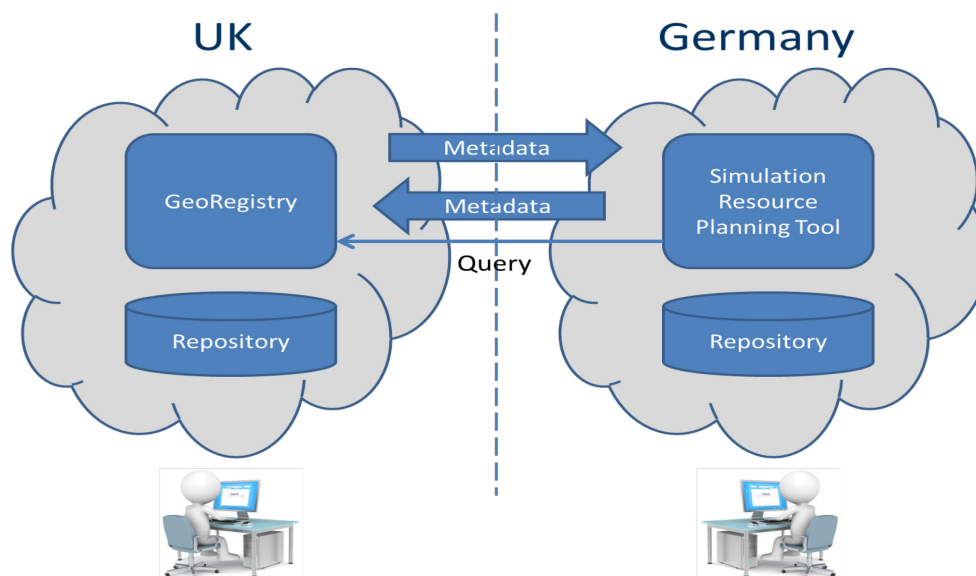


Figure 3-7: The Ultimate Goal of the Work Carried Out in This Experiment is the Exchange of Data Between the UK and German Registry-Repository Capabilities.

3.4.5 Processes and Activities

The experimentation is split into two phases. During Phase 1, the following activities will be carried out:

- Evaluate MSG-136 Service Specification Template:
 - Map MSG-136 Service Description Template to AIMS metadata:
 - Compare the two approaches and produce a high-level mapping between them;
 - Document any differences; and
 - Document issues/limitations/gaps.
 - Document any limitations with MSG-136 Service Description Template.

The intention during Phase 2 is to carry out activities involving machine-to-machine communication between the UK registry and Aditerna software as follows:

- Evaluate the ability to exchange metadata between Registries from different nations:
 - Manually harvest service(s) defined by MSG-136 Service Description Template into GeoRegistry;
 - Define above service(s) using AIMS metadata and manually harvest into GeoRegistry;
 - Explore different ways of Discovering M&S Resource using both metadata descriptions of M&S Resource; and
 - Document issues/limitations.
- Evaluate ability to Discover and obtain M&S Resources (Service/Data/Asset) using Registries/Repositories from different nations:
 - Perform Human-Machine Discovery of M&S Resources using GeoRegistry and SRP Tool;
 - Discover M&S Resource(s) common to both GeoRegistry and SRP Tool;
 - Document differences/limitations/issues/gaps; and
 - Use SRP Tool to perform a machine-machine query on GeoRegistry.

The ability to perform this second phase in full is limited by the lack of an API for the Aditerna software. Without this, no machine-to-machine communication is available, exchange of data is limited to manual additions, and searches are limited to those that can be performed via its graphical user interface. Tasks that could still be carried out within the list above are highlighted in bold; these do not involve machine-to-machine communication, just manual harvesting. However, it isn't clear what benefits these tasks would provide to MSG-136 in isolation. They were originally intended as test cases for the automated work that was to be completed later on. In themselves, they merely demonstrate in more detail what we have already shown by performing the mapping and review of the Service Specification Template included later in this document.

In order to implement the second phase of experimentation as envisaged, a complete and documented external API would be required for the Aditerna software.

It is suggested that as an alternative, time could be spent on developing the material produced during Phase 1 to document the two alternative approaches, and understanding the differences in more detail. In particular, the mapping produced that shows the relationship between the SST and the UK information model components could be of use in understanding the differences in the two approaches as well as the areas in which the two different approaches complement each other.

3.4.6 Observations

These are recorded in Section 5.4 and Appendix 5-1 of TR-MSG-136-Part-V, Modelling and Simulation as a Service, Volume 2: MSaaS Discovery Service and Metadata [2]. Appendix 5-1 provides a mapping of the MSG-136 Service Descriptions with the UK AIMS metadata.

3.4.7 Outcome/Analysis

N/A

3.5 TEST CASE 4: SIMULATION COMPOSITION AND DEPLOYMENT (GBR)

3.5.1 Objective/Topic/Question

- Determine how ‘Service Compositions’ can be stored and discovered.
- Determine how services can be composed.
- Determine how ‘Service Deployments’ can be stored and discovered.
- Determine how to deploy the same simulation deployment for concurrent multiple events.

3.5.1.2 Scope

ABBs to be evaluated:

- M&S Composition Services.

3.5.2 Assumptions/Preconditions/Boundary Conditions

- GeoRegistry operating in a cloud environment.
- Metadata available for simulation resources.

3.5.3 Systems and Interfaces

- Systems
 - UK GeoRegistry.
 - HTTP Query Tools (COTS).
 - Infrastructure as Code Tools (COTS).
 - Prototype Composer Tool.
 - Prototype Deployment Tool.
- Interfaces
 - GeoRegistry: ebRIM.
 - HTTP/HTTPS.
 - Web Processing Service (WPS).
- Data

- Example metadata for Simulation Compositions, Services and Deployment objects.
- Example Information Objects for Compositions and Services.
- GeoRegistry/Repository populated with example M&S Simulation Compositions and Simulation Deployments.

3.5.4 Test Setup

Figure 3-8 provides an overview of the experimental setup.

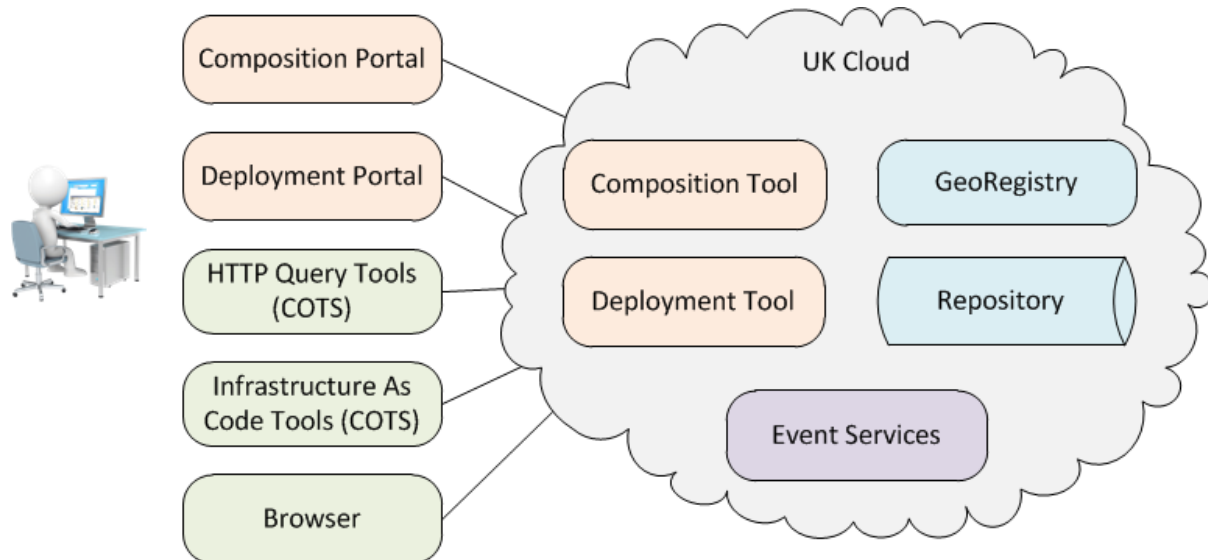


Figure 3-8: Experimental Setup.

3.5.4.1 Definitions

A list of definitions is provided in Table 3-3.

Table 3-3: List of Definitions.

Annotation	Supporting documents or data used to evaluate a resource. An annotation is stored in the Registry's Repository whereas other supporting documents are stored in an external repository.
Composition	A set of services which work together to provide a simulation environment for a defined event.
Event	An instance of the execution of a particular simulation deployment.
External Link	A link to supporting documents or data used to evaluate a resource. An external link is stored in an external repository.
Sub-composition	A group of re-usable services that have been composed and integrated and can be readily deployed on-demand, as a composition or as part of a larger composition.

3.5.4.2 Composition

Test #1: Determine how can ‘Service Compositions’ be stored and discovered:

- Evaluate different types of complex queries for discovering Simulation Compositions and sub-Compositions;
- Produce sub-Compositions;
- Define additional attributes for a ‘Service Composition’ in the MSaaS information model; and
- Document approach.

Test #2: Determine how services can be composed:

- Develop and evaluate different approaches to service composition;
- Develop prototype Composition tool to concatenate sub-compositions; and
- Document approach.

3.5.4.3 Deployment

Test #3: Determine how ‘Service Deployments’ can be stored and discovered:

- Perform Human-Machine and Machine-Machine queries to Discover Simulation Deployments;
- Deploy ‘Infrastructure as Code’ for stored example ‘Service Deployments’ in the Registry/Repository;
- Define additional attributes for a ‘Service Deployment’ in the MSaaS information model; and
- Document approach.

Test #4: Determine how to deploy the same simulation deployment for concurrent multiple events:

- Determine approach;
- Develop prototype Simulation Deployment tool for controlling/monitoring Simulation Deployments;
- Enhance prototype Simulation Deployment tool for deploying concurrently running deployments;
- Demonstrate simultaneous execution of Simulation Deployments; and
- Document approach.

3.5.4.4 Composition Concepts

The proposed sub-composition concept is to describe a group of re-usable services that have been composed and integrated and can be readily deployed on-demand, as a composition or as part of a larger composition.

A composition is a set of services which work together to provide a simulation environment for a defined event. It should be noted that the structure of a Composition and sub-Composition Resource Information object is the same i.e., there is no master Resource Information Composition object. An example composition is shown in Figure 3-9.

Sub-Compositions can be overlaid onto the previous diagram to show the concept of grouping services in a verified sub-composition.

In Figure 3-10, the ‘Exercise Control’ composition is a group of integrated services providing the capability to control an exercise from a single Reconfigurable User Interface (RUI).

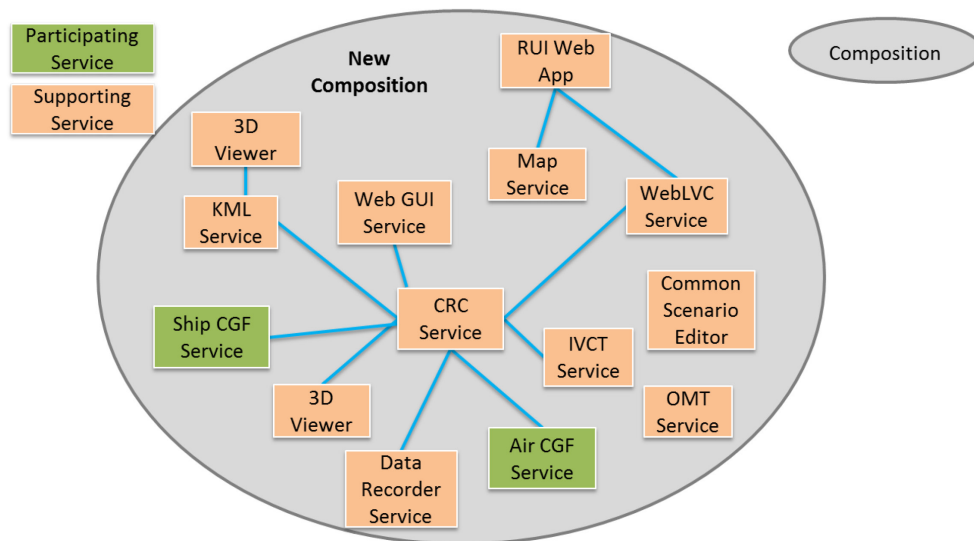


Figure 3-9: Example Composition.

Each service is described as an individual supporting service within the registry, but has been pulled together into a single sub-composition, to be used in combination with other sub-compositions to meet an Event's requirement.

An Event is an instance of the execution of a particular simulation deployment.

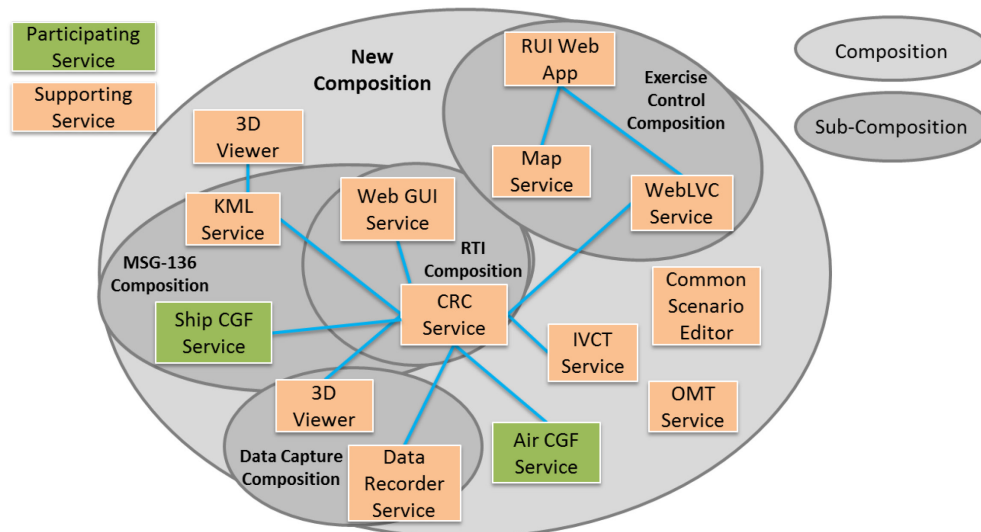
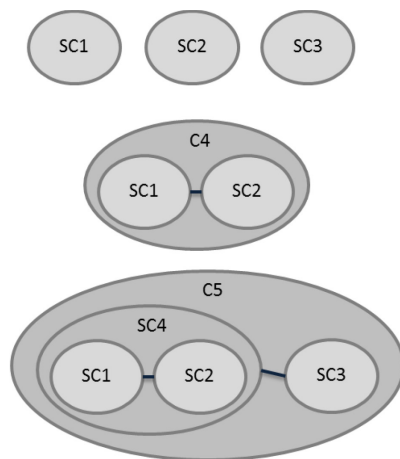


Figure 3-10: Sub-Composition Example.

3.5.4.4.1 Defining Sub-Compositions

How a group of integrated services are divided into sub-compositions needs to be carefully considered in order to provide the best reuse potential. They can be integrated in a hierarchical manner as shown in Figure 3-11.



- Sub-Compositions are integrated services created by simulation developers or service providers
- Composition 4 is comprised of sub-compositions 1 and 2
- Composition 4 is re-used, as a sub-composition in Composition 5

Figure 3-11: Sub-Composition Re-Use.

3.5.4.4.2 Deployment Concepts

The proposed deployment concept is to support automation of deployments by describing deployments as code.

Previous research has highlighted the difficulties of porting virtual machines between different clouds (although common standards were used). As a result, it is recommended that the use of ‘Infrastructure as Code (IaC)’ is used for defining how the services should be deployed for a simulation.

Infrastructure as Code enables infrastructure to be provided on demand. Infrastructure is treated as code and uses templates and a descriptive language to define the infrastructure.

These templates are used to automate the creation and setup of the infrastructure, providing consistency, repeatability and removal of manual errors. This also allows infrastructure elements to be easily shared and reused between different cloud environments.

3.5.4.4.3 MSaaS Information Model

In Figure 3-12, the three-layered MSaaS Information Model is made up of a Registry, Metadata and an Information layer.

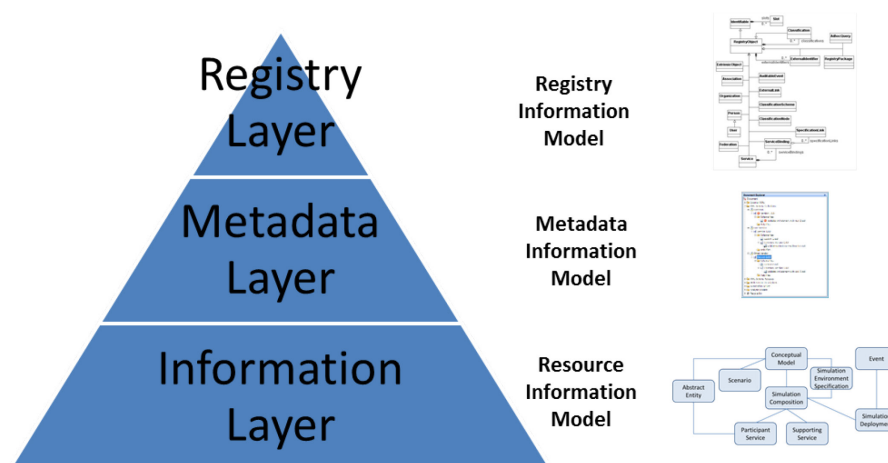


Figure 3-12: MSaaS Information Model.

The layers support the ecosystem processes for searching, discovering, evaluating and automating, as follows:

- a) The registry layer to search for, and discover resources within the Ecosystem;
- b) The metadata layer to evaluate the suitability of resources against the simulation requirements; and
- c) The Information layer contains Information Objects that provide additional information specific to a Registry object that can extend the search, discovery, evaluate and automation capabilities.

The process for reusing simulation resources is to discover them, evaluate their suitability and to use them as shown in Figure 3-13.

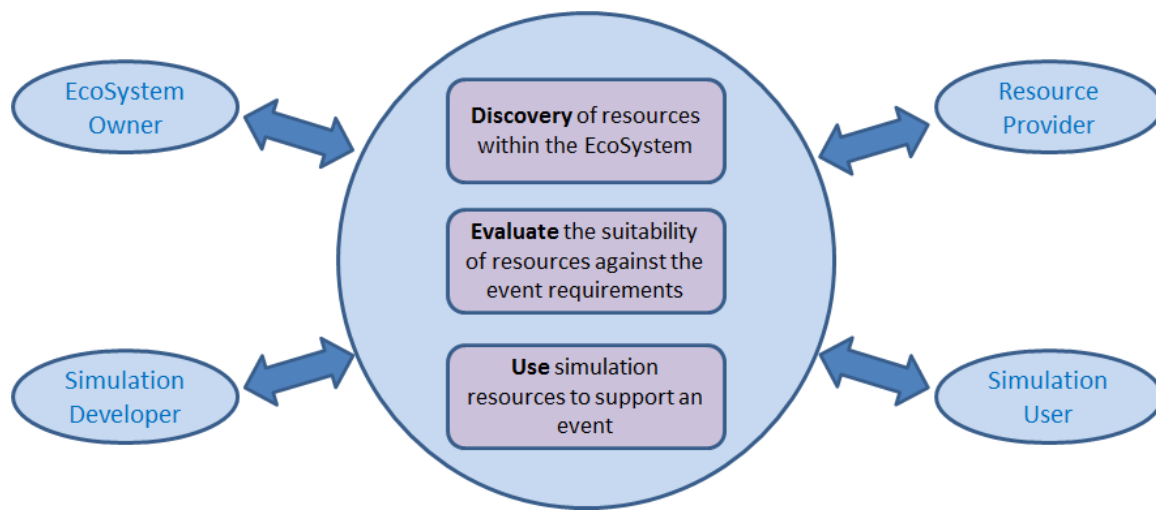


Figure 3-13: Discover, Evaluate and Use.

3.5.4.4.4 *Supporting Registry Tools*

To support the process of harvesting objects into the registry, the following tools have been developed and deployed by Envitia:

- 1) An FTP Server was setup on the GeoRegistry VM on the UKCloud. This allows developers to upload metadata, annotations, associations and information objects to the staging area, ready for harvesting.
- 2) A supporting Harvesting Tool was developed to allow developers to harvest metadata into the Registry.
- 3) A supporting Annotation Tool was developed to allow developers to load annotations into the Registry's Repository.
- 4) A supporting Association Tool was developed to allow developers to load associations between objects in the Registry.
- 5) A supporting Information Object Tool was developed to allow developers to load information objects into the Registry.

These tools were successfully used to harvest the metadata, load associations and information objects for the services, compositions, sub-compositions and deployments required for these experiments as shown in Figure 3-14.

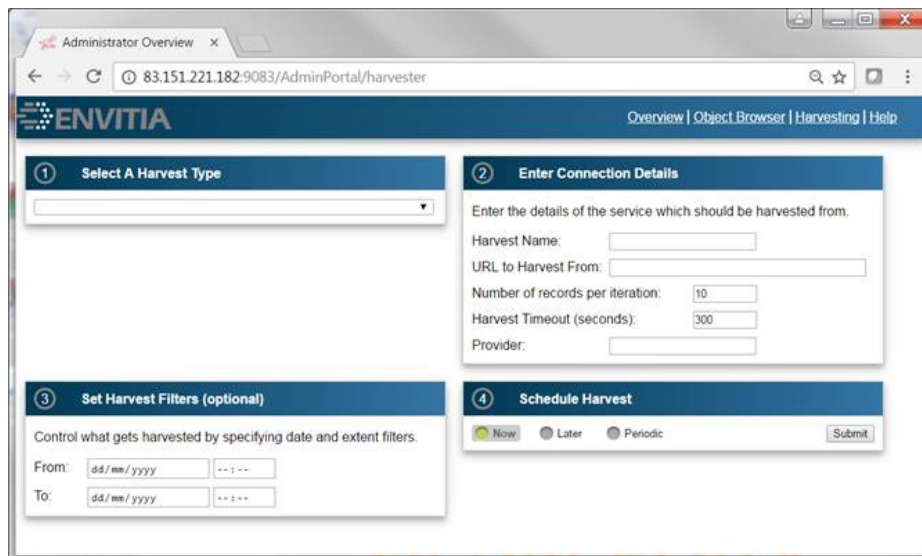


Figure 3-14: Harvesting Tool.

3.5.5 Processes and Activities

3.5.5.1 Test #1

3.5.5.1.1 Test #1 Experimental Setup

This test case evaluated different types of complex queries for discovering simulation compositions and sub-compositions. Example sub-compositions were produced as Information Objects that could be harvested into the Registry's Repository, as shown in Figure 3-15.

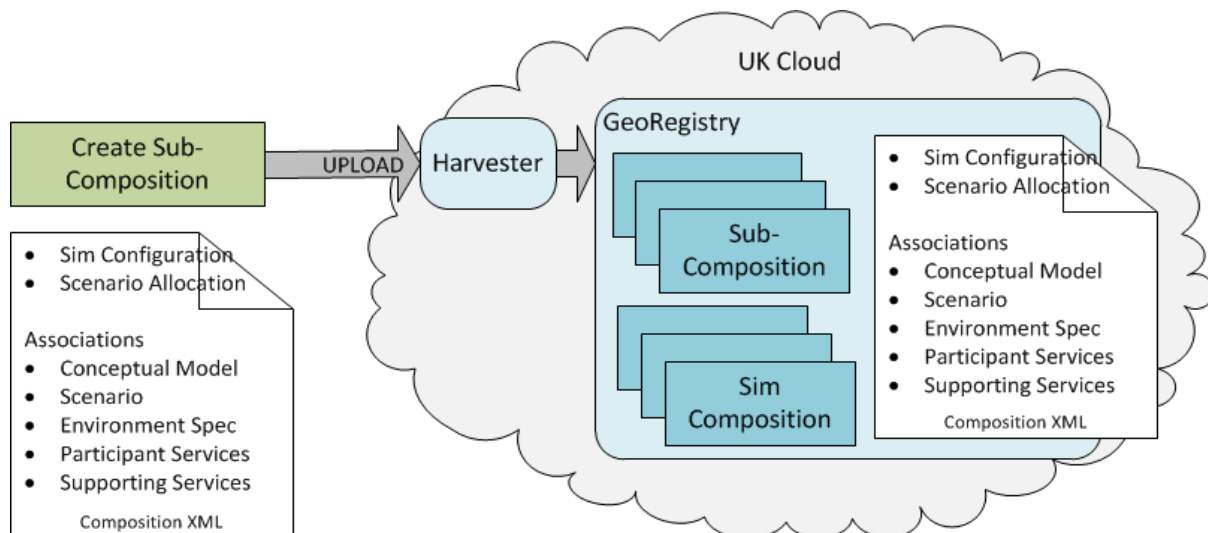


Figure 3-15: Create Composition Process.

3.5.5.1.1.1 Sub-Compositions

Figure 3-16 shows one of the compositions (and sub-compositions) added to the registry to support this experiment.

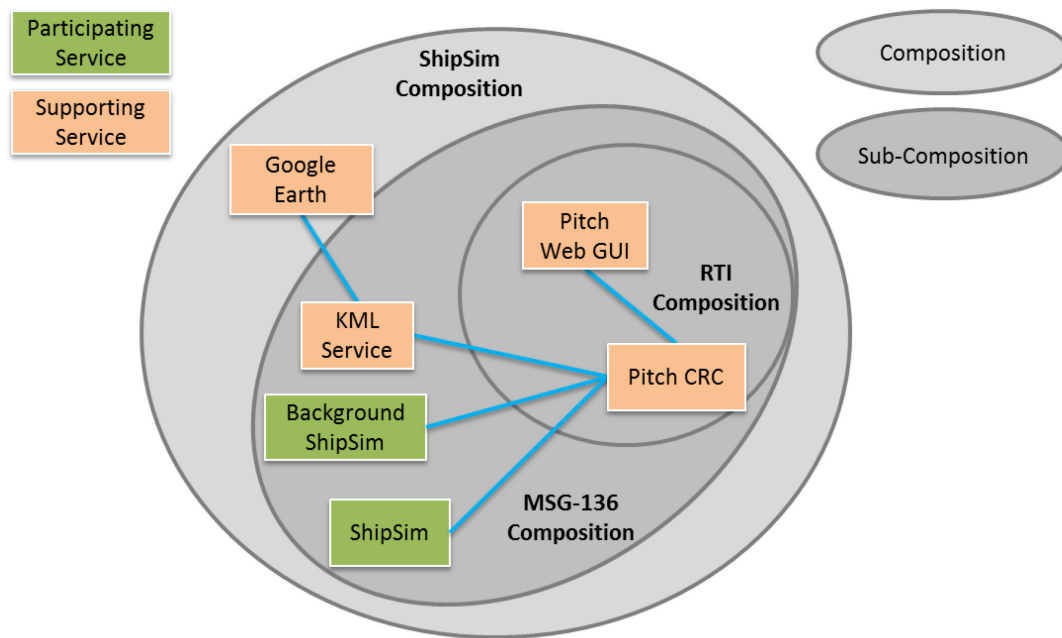


Figure 3-16: Experiment Composition.

A prototype HTTP Client Web tool was developed as part of this experiment, which was used to query the registry and display the results as shown in Figure 3-17. The prototype Web tool was developed using the Java Spring framework, with an Angular front end.

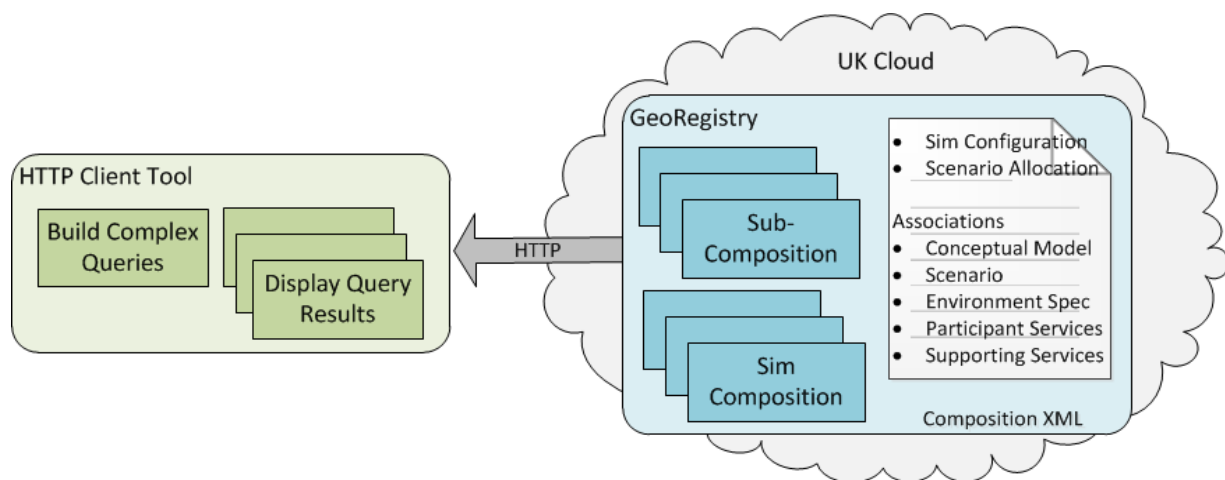


Figure 3-17: Prototype HTTP Client Web Tool.

3.5.5.1.1.2 Storing Sub-Compositions

Example compositions (and Sub-compositions) were created as XML files. Metadata files and association files were manually created to describe the compositions and define the relationships between them.

The supporting registry tools described earlier, were used to harvest compositions, sub-compositions into the registry.

3.5.5.1.1.3 Complex Queries

A complex query combines a number of simple queries in order to discover compositions that meet an Event's requirements. They exploit one of the benefits of the Registry Information Model. This task explored the types of complex queries required to discover compositions.

These complex queries supported the processes to:

- a) Search and discover resources;
- b) Evaluate resources; and
- c) Automate (or semi-automate) composition of simulations.

3.5.5.1.1.4 Search and Discovery

When searching for, and discovering resources within the Ecosystem, the resources are matched against a required capability, a characteristic or a specific resource. Information that supports search and discovery is stored as metadata, keywords, associations or classifications within the registry.

Example queries include:

- a) Search for events in the Air domain;
- b) Search for all scenarios based in the UK;
- c) Search for scenarios created by me;
- d) Search for compositions using HLA-Evolved with NETN FOM;
- e) Search for services delivering Close Air Support (CAP) capability;
- f) Search for services delivering terrain databases in the UK;
- g) Search for deployments using the UKCloud infrastructure;
- h) Search for deployments associated with <event id>; and
- i) Search for events used for Test and Evaluation.

3.5.5.1.1.5 Evaluate

When evaluating the suitability of resources, the queries performed provide the user with additional data in order to assess the resource against the simulation event requirements.

This information is stored as a combination of metadata, associations and supporting documents or data within the registry.

Additional supporting information is stored as either an annotation or as an external link. These are documents or data that can be used to evaluate the resource, for example, a video of a scenario, an interface control document or simulation developers' experience of using a resource.

Examples of queries include:

- a) Evaluate the compositions using <scenario id>;
- b) Evaluate the data services delivering terrain databases in the UK;
- c) Evaluate the capabilities provided by <service id>;
- d) Evaluate the interfaces provided by <service id>;

- e) Evaluate the information captured for <event id>;
- f) Evaluate the lessons learnt captured for <event id>; and
- g) Evaluate the constraints of <composition id>.

3.5.5.1.1.6 Automation

One of the goals of MSaaS is to provide as much automation as possible to the process of creating, composing and deploying simulations.

When automating the composition, multiple queries are required that traverse the Registry's Resource Information model. These combined queries perform the search and discover functions that return information to be evaluated. The evaluation is currently performed by a human and enables a decision to be made as to the suitability of a simulation resource. The resulting decision selects a suitable combination of resources that best meet the event requirements.

The 'decide' function is not provided by the registry, but requires a management layer or suite of automation tools to support this, which use the power of the registry.

The proposed concept is to:

- a) Combine services into sub-compositions, in order to pull in a group of integrated services providing a required capability, for example 'Exercise Control', or 'Data Capture'.
- b) Apply rules to decide on a suitable combination of resources and compositions which best meet the simulation event requirements, for example, select services with the highest user rating, or select services which can be deployed within a required budget.

Sub-compositions combine services, as a group of re-usable services that have been composed and integrated and can be readily deployed on-demand.

These sub-compositions can then be used by automation tools to pull together a suitable combination of resources as part of a larger composition, which best meet the requirements.

Example requests include:

- a) Build me a simulation for a Test and Evaluation event using <service id>;
- b) Build me a simulation for a collective training event, to support a joint 'Tactical Engagement' exercise, using <conceptual model id> and <scenario id>;
- c) Build me a simulation for specific simulator training using <simulator id>;
- d) Build me a simulation based on <event id> using <scenario id> with the capability to capture and analyse the data;
- e) Build me a simulation using a 'desert' terrain environment to support a land exercise; and
- f) Build me a simulation using <service id> and <service id> with HLA-E and NETN FOM.

3.5.5.1.2 Test #1 Observations

3.5.5.1.2.1 Search and Discovery

The prototype HTTP Client Web tool can be used to search for, and discover resources within the Ecosystem, as shown in Figure 3-18.

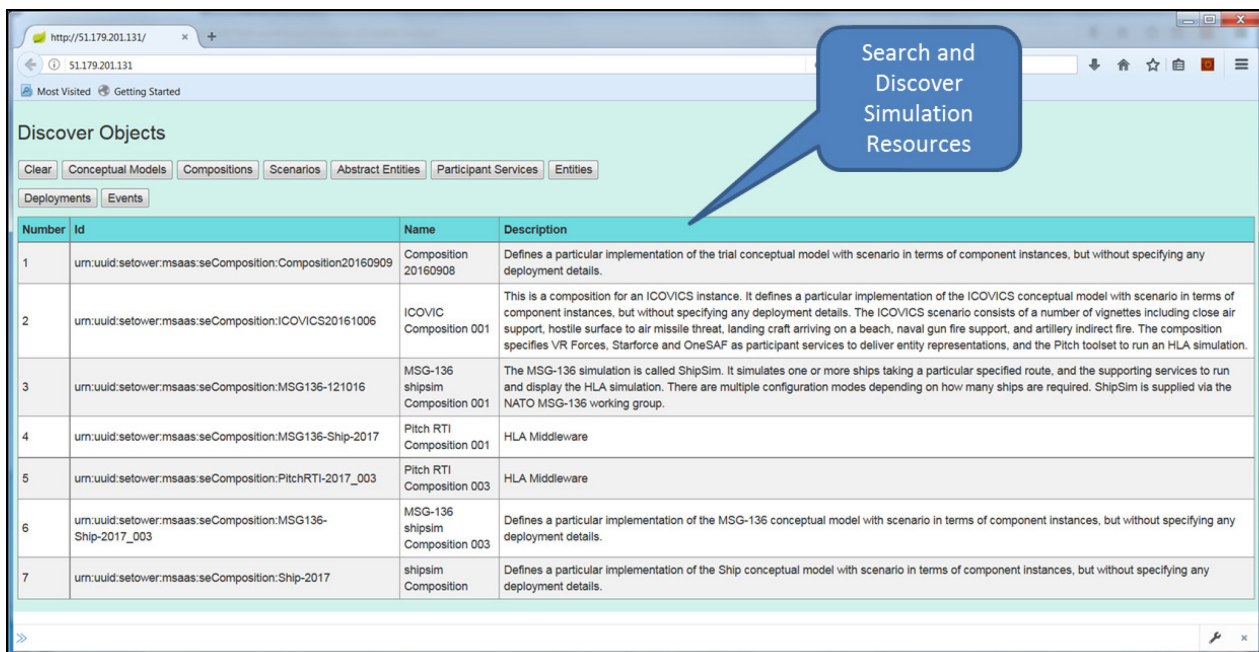


Figure 3-18: Search and Discovery using HTTP Client Tool.

3.5.5.1.2.2 Evaluate

The prototype HTTP Client Web tool can be used to perform some simple evaluation based on associations between resources, as shown in Figure 3-19.

Ideally this would also cover additional evaluation against metadata, annotations or external links. This capability does not form part of the experiment.

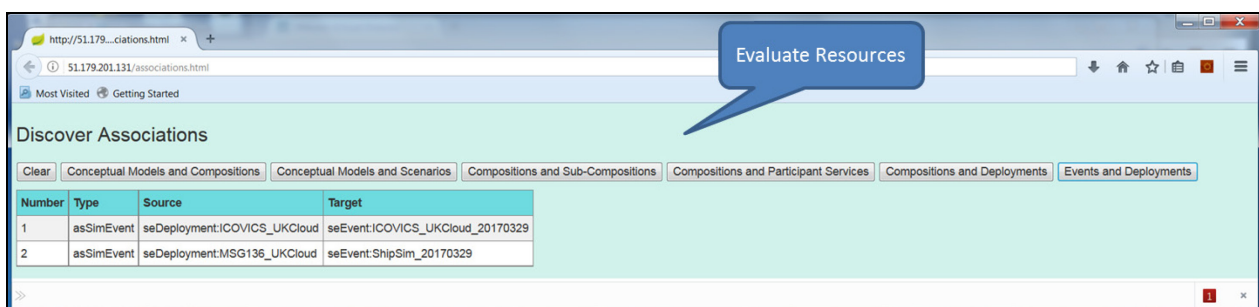


Figure 3-19: Evaluate Associations Using HTTP Client Tool.

3.5.5.1.2.3 Automation

The prototype HTTP Client Web tool can be used to perform a combination of queries. These queries can be used by automation tools to perform the search, discover and evaluation tasks, as shown in Figure 3-20.

Sub-compositions can be used to pull in a group of integrated services providing a required capability, for example 'Exercise Control', or 'Data Capture', as shown in Figure 3-21.

Rules can be applied by an automation tool to decide on a suitable combination of resources which best meet the event requirements, for example, select services with the highest user rating, or select services which can be deployed within a required budget.

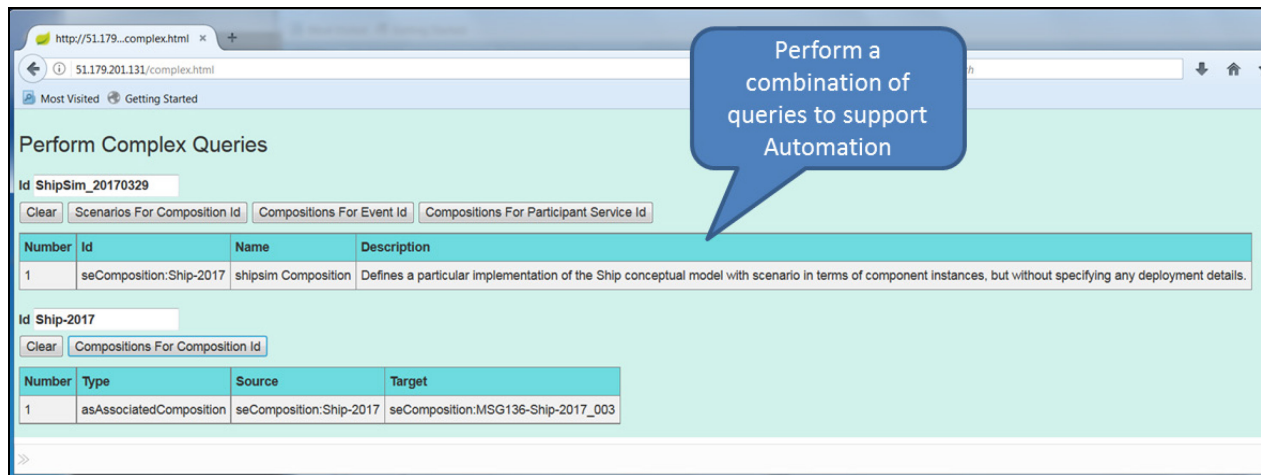


Figure 3-20: Automate Using HTTP Client Tool.

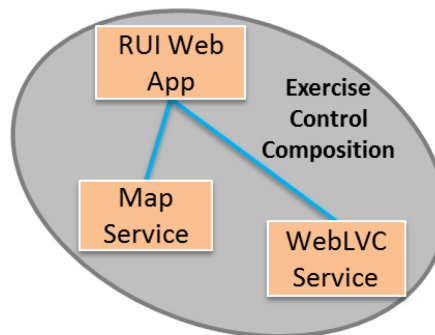


Figure 3-21: Exercise Control Sub-Composition.

3.5.5.1.3 Test #1 Outcome/Analysis

The objective was to determine how 'Service Compositions' can be stored and discovered.

The experiment has:

- Developed the composition (and sub-composition) concept;
- Reviewed and extended the MSaaS Information model to support composition concepts;
- Developed additional supporting tools to allow developers to harvest into the registry; and
- Demonstrated how complex queries can be used to support automation of composition.

The test has shown that semi-automation of compositions for a simulation should be possible. Combining a number of queries is the first step towards this goal.

There is a need for a suite of composition automation tools to be integrated with the registry, in order to succeed in simplifying the complexity of composing an event.

3.5.5.2 Test #2

Determine how services can be composed using the event requirements to search the registry for candidate services, sub-compositions and compositions.

The focus is on compositions created by the simulation user e.g., Front Line Commands, using existing services, sub-compositions and compositions, as opposed to new services or sub-compositions created by the simulation developer or service provider.

3.5.5.2.1 Test #2 Experimental Setup

This test case evaluated different approaches to service composition and developed a prototype composition tool to concatenate discovered sub-compositions.

This task explored options to concatenate sub-compositions discovered via complex queries. The issues explored included:

- Can event requirements be used to create a composition?
- What information can be used to decide which sub-compositions to use?
- How can requirement gaps be resolved?
- Can rules be used to apply supporting sub-compositions?
- How can the selection/choices be presented to the user?

In order to explore these questions, a prototype composer tool and Web Processing Service (WPS) were developed to incorporate querying the registry and display the results (see Figure 3-22).

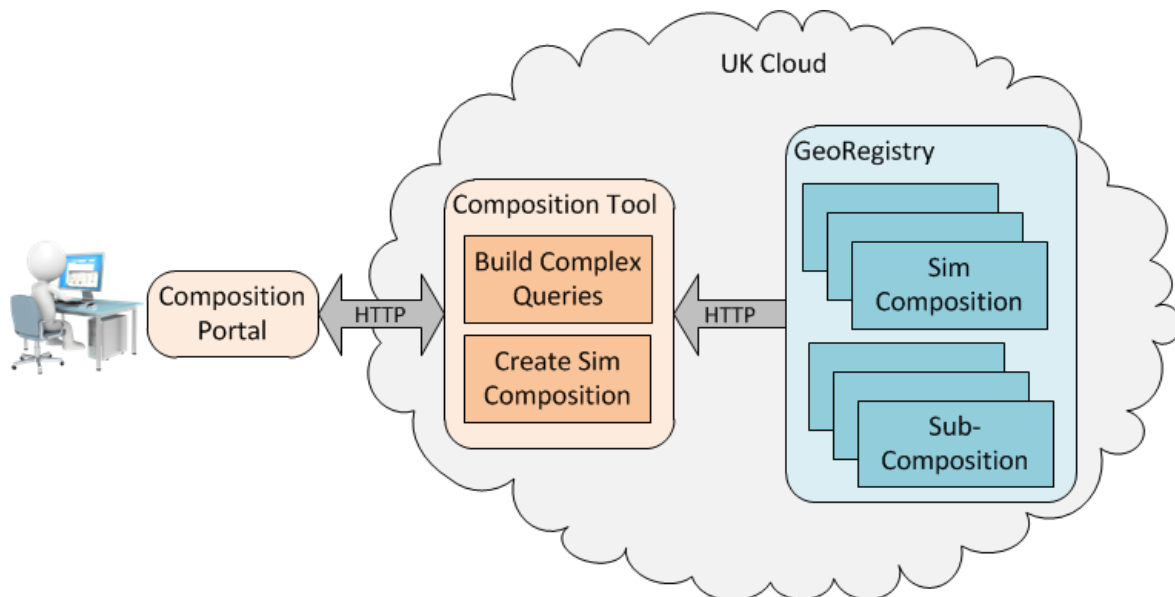


Figure 3-22: Composer Tool.

Example Information Objects that described services and compositions (and sub-compositions) were created as JSON files. The supporting registry tools described earlier were used to harvest these Information Objects into Information layer of the registry.

3.5.5.2.2 *Test #2 Observations*

3.5.5.2.2.1 *Composition Web Processing Service (WPS)*

The Composer WPS uses complex queries to identify matching or closely matching (ranked) compositions (which also include supporting services) from user defined system requirements, as shown in Figure 3-23.

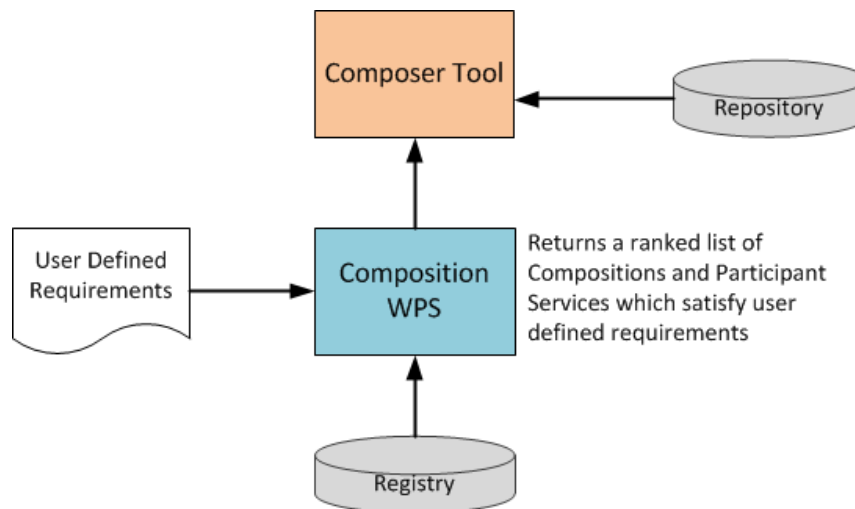


Figure 3-23: Composition WPS.

The WPS could also apply rules to decide on a suitable combination of resources which best meet the event requirements, for example, select services with the highest user rating, or select services that can be deployed within a required budget. This has not been implemented for this initial version of the WPS.

For this experiment, the user defined requirements were described in terms of one or more of the following:

- 1) Conceptual Entity:
 - a) Entity Type e.g., UAV.
 - b) Property e.g., Range.
 - c) Behaviour e.g., Follow.
- 2) Environment Entity:
 - a) Entity Type e.g., Map.
 - b) Property e.g., WMS.
- 3) Support Capability:
 - a) Property:
 - i) Function e.g., Exercise Control.
 - ii) Type e.g., Rich Internet App.
 - iii) Protocol e.g., HLA-E.
 - iv) FOM e.g., NETN v1

3.5.5.2.2.2 Composer Tool

The Composer tool is shown in Figure 3-24. The tool queries the WPS with a pre-defined user requirement and displays the results.

The table on the left-hand side is a ranked list of compositions and services, returned from the WPS, which match or partially matches the requirements fed into the WPS.



The screenshot shows the MSaaS Composer interface. On the left, there is a table with the following data:

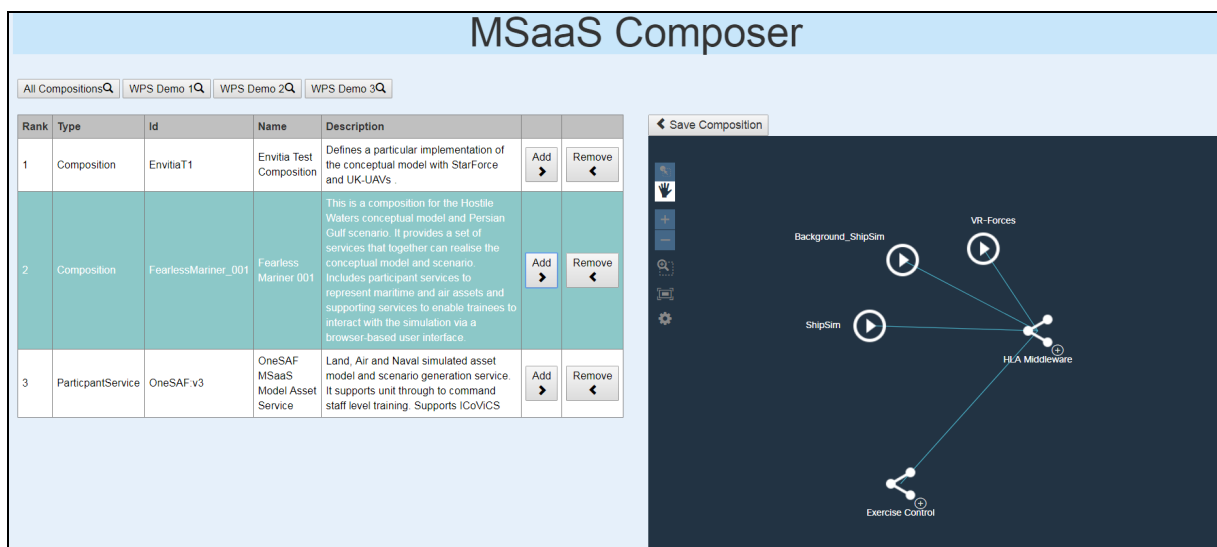
Rank	Type	Id	Name	Description	Add	Remove
1	Composition	FearlessMariner_001	Fearless Mariner 001	This is a composition for the Hostile Waters conceptual model and Persian Gulf scenario. It provides a set of services that together can realise the conceptual model and scenario. Includes participant services to represent maritime and air assets and supporting services to enable trainees to interact with the simulation via a browser-based user interface.	➤	➤
2	Composition	EnvitiaT1	Envitia Test Composition	Defines a particular implementation of the conceptual model with StarForce and UK-UAVs.	➤	➤
3	ParticipantService	OneSAF.v3	OneSAF MSaaS Model Asset Service	Land, Air and Naval simulated asset model and scenario generation service. It supports unit through to command staff level training. Supports ICoVICS	➤	➤

On the right side of the interface, there is a 'Save Composition' button and a graphical topology view showing a network of nodes and connections.

Figure 3-24: MSaaS Composer.

Each item (Composition or Service) can be added (or removed) from/to the topology view (displayed on the right-hand side). When an item is added, the Composer Tool queries the registry for the Information Object associated with the selected Composition or Service, and displays a graphical illustration of this object as a number of linked nodes in the topology view (see Figure 3-25).

For a composition (or sub-composition) this shows the services that make up the composition and how they are associated.



The screenshot shows the MSaaS Composer interface with the graphical topology view. The table on the left is the same as in Figure 3-24. The topology view on the right shows a network of nodes and connections. The nodes are labeled: Background_ShipSim, VR-Forces, ShipSim, HLA Middleware, and Exercise Control. The connections are as follows:

- Background_ShipSim is connected to VR-Forces.
- VR-Forces is connected to ShipSim.
- ShipSim is connected to HLA Middleware.
- HLA Middleware is connected to Exercise Control.

Figure 3-25: MSaaS Composer, with Graphical Illustration.

Each sub-composition is initially shown as a single node (collapsed view). By selecting the + (plus) icon, the sub-composition is expanded to show its contents. For example, the expanded sub-composition in Figure 3-26 shows that ‘Exercise Control’ consists of the following services:

- a) WebLVC Server;
- b) GeoServer;
- c) ICoVICS RUI; and
- d) ArcGIS Online.

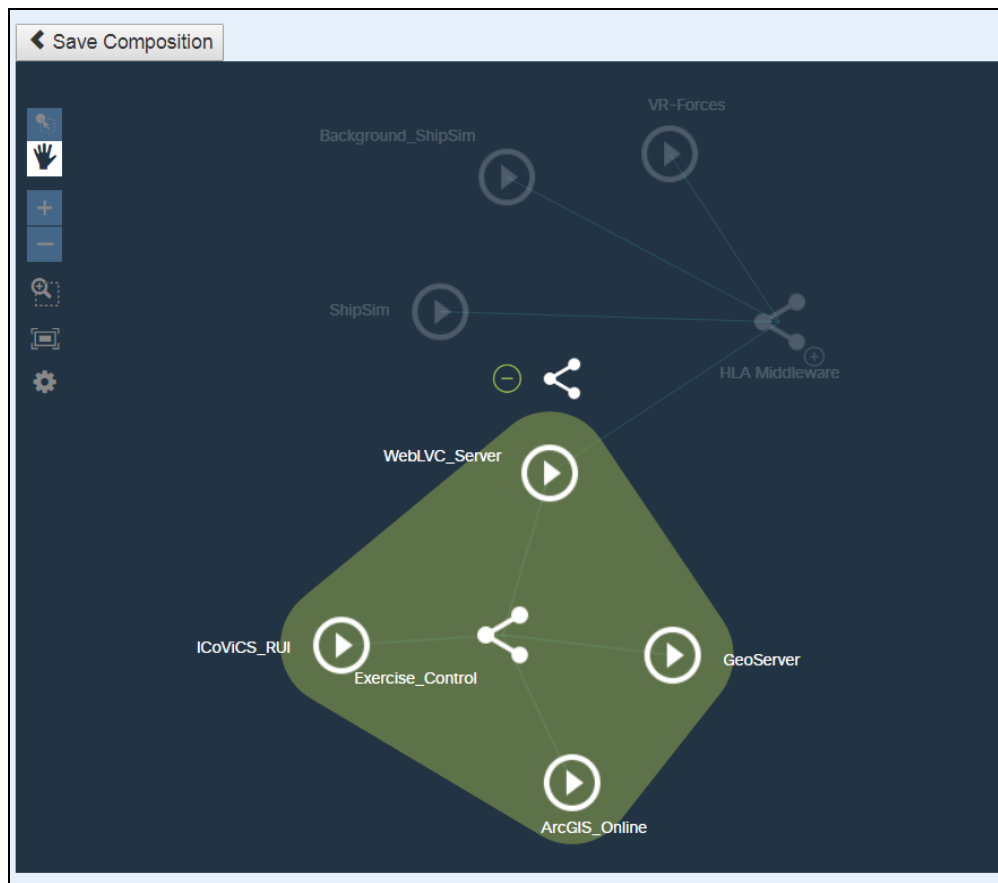


Figure 3-26: Topology View.

The Topology view can be added to, by selecting additional results from the list and adding them to the topology view. In Figure 3-27, OneSAF Service has been added.

As shown in Figure 3-28, the new Service (OneSAF) can then be linked to (associated with) another Service (Pitch RTI), ready to be saved as a new composition. In the current implementation no check is made as to whether the new service can viably be integrated with the selected node.

The prototype composer provides basic functionality e.g., collapsed/expanded sub-compositions, individual services and links (associations) can be removed or added, nodes can be renamed.

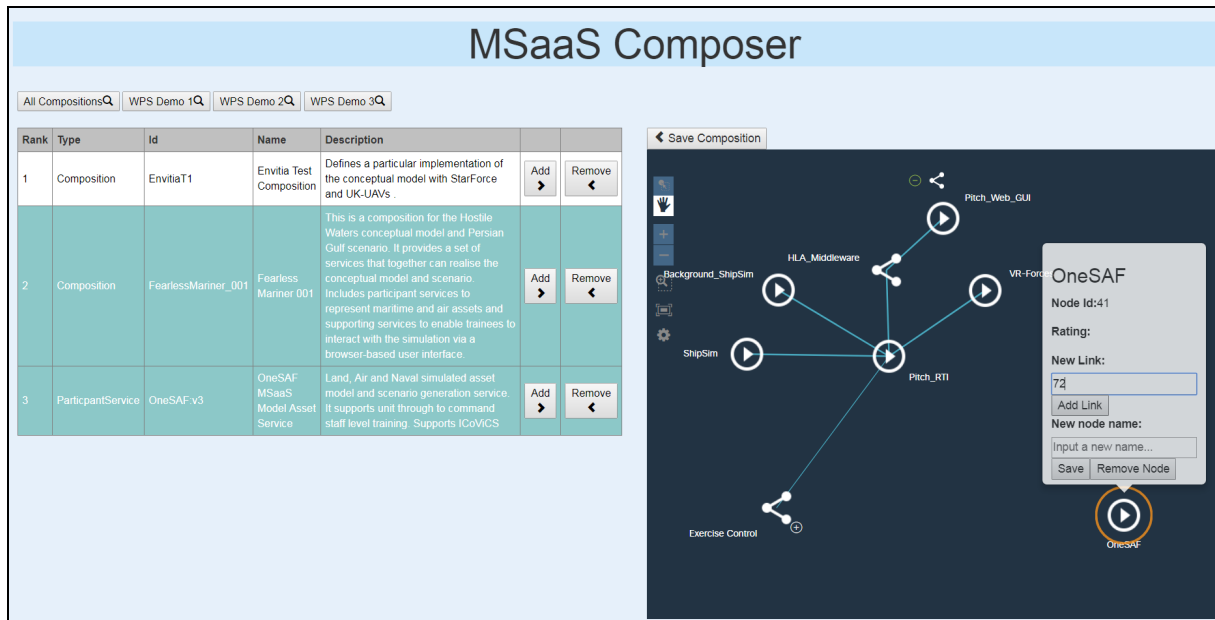


Figure 3-27: MSaaS Composer.

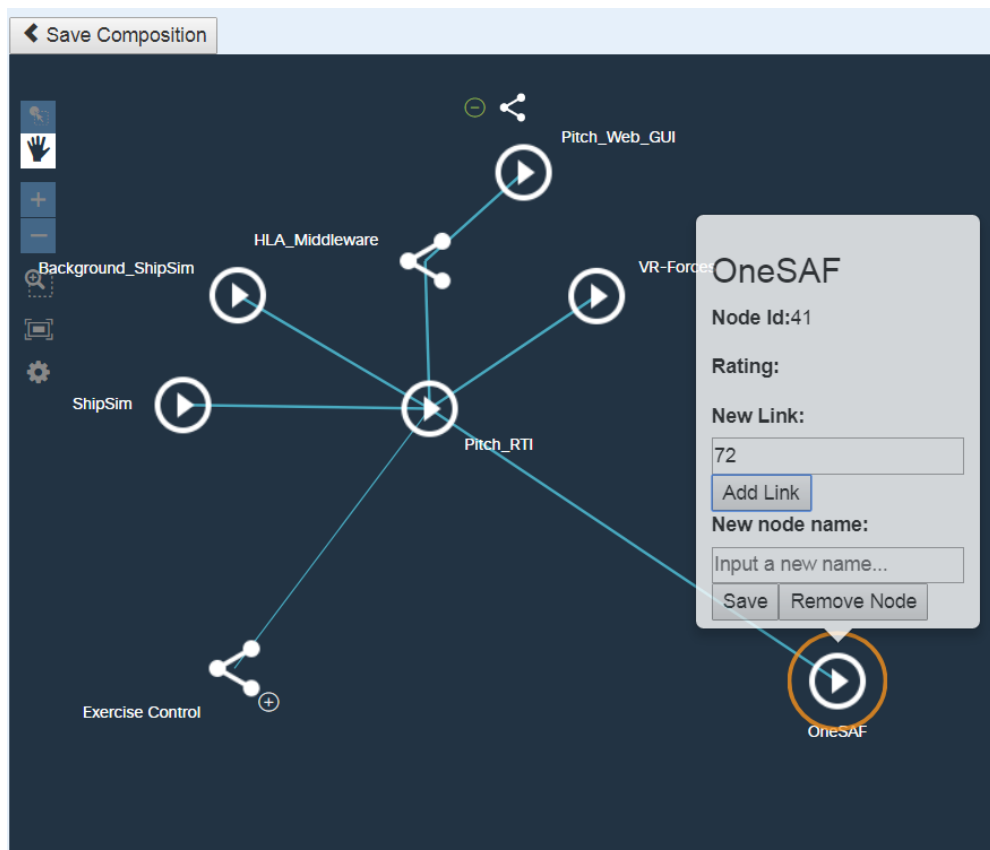


Figure 3-28: Topology View.

3.5.5.2.3 Test #2 Outcome/Analysis

The objective was to determine how services can be composed.

This experiment has:

- Developed a Composition Web Processing Services (WPS) to identify matching or closely matching (ranked) compositions (which also include supporting services) from user defined system requirements.
- Developed a prototype Composer tool to graphically view selected sub-compositions and services.

There is a need for a suite of composition automation tools to be integrated with the registry, in order to succeed in simplifying the complexity of composing an event.

This prototype tool has allowed the concept to be explored and enabled requirements for such a tool to be developed and understood.

3.5.5.3 Test #3

Determine how ‘Service Deployments’ can be stored and discovered.

3.5.5.3.1 Test #3 Experimental Setup

This test case created and deployed Infrastructure as Code for stored example ‘Service Deployments’ in the Registry/Repository, as shown in Figure 3-29.

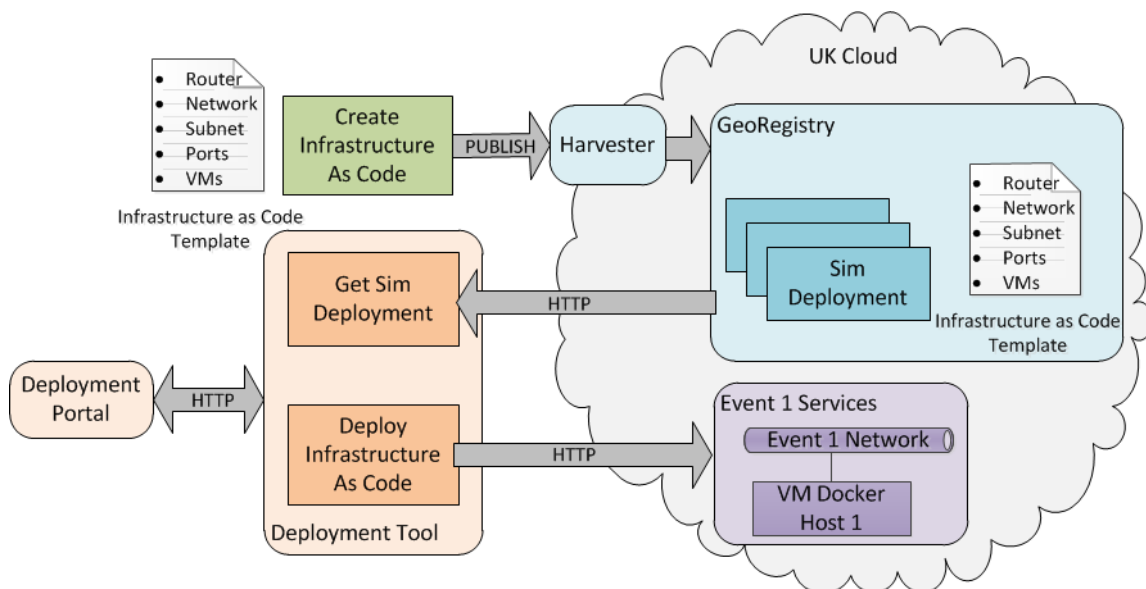


Figure 3-29: Experiment Setup.

A prototype deployment tool has been developed to query the registry for available deployments and deploy events, which are described using Infrastructure as Code.

3.5.5.3.1.2 Storing Deployments

Example deployments were created as XML files using a text editor. Metadata files and association files were then manually created to describe the deployments and define the relationships to other objects in the registry.

The supporting registry tools described earlier were used to harvest deployments into the registry.

3.5.5.3.1.2 Infrastructure as Code

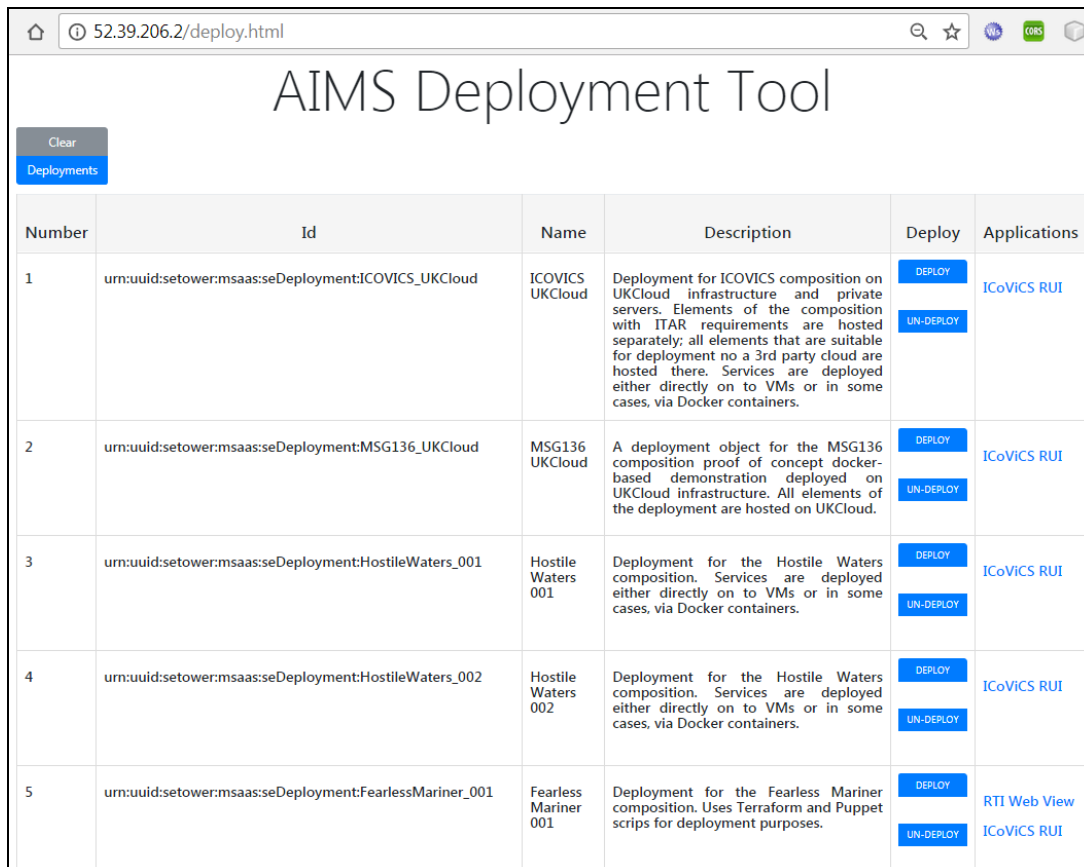
Infrastructure as Code enables infrastructure to be provided on demand. Infrastructure is treated as code and uses templates and a descriptive language to define the infrastructure.

These templates are used to automate the creation and setup of the infrastructure, providing consistency, repeatability and removal of manual errors. This also allows infrastructure elements to be easily shared and reused between cloud environments.

3.5.5.3.2 Test #3 Observations

3.5.5.3.2.1 Deployment Tool

The Deployment Tool is shown in Figure 3-30. The tool queries the registry for all deployments and allows the user to deploy or un-deploy each event. Each event may have a number of links to applications available to the user, which have been deployed for this event.



Number	Id	Name	Description	Deploy	Applications
1	urn:uuid:setower:msaas:seDeployment:ICOVICS_UKCloud	ICOVICS UKCloud	Deployment for ICOVICS composition on UKCloud infrastructure and private servers. Elements of the composition with ITAR requirements are hosted separately; all elements that are suitable for deployment no a 3rd party cloud are hosted there. Services are deployed either directly on to VMs or in some cases, via Docker containers.	DEPLOY UN-DEPLOY	ICoViCS RUI
2	urn:uuid:setower:msaas:seDeployment:MSG136_UKCloud	MSG136 UKCloud	A deployment object for the MSG136 composition proof of concept docker-based demonstration deployed on UKCloud infrastructure. All elements of the deployment are hosted on UKCloud.	DEPLOY UN-DEPLOY	ICoViCS RUI
3	urn:uuid:setower:msaas:seDeployment:HostileWaters_001	Hostile Waters 001	Deployment for the Hostile Waters composition. Services are deployed either directly on to VMs or in some cases, via Docker containers.	DEPLOY UN-DEPLOY	ICoViCS RUI
4	urn:uuid:setower:msaas:seDeployment:HostileWaters_002	Hostile Waters 002	Deployment for the Hostile Waters composition. Services are deployed either directly on to VMs or in some cases, via Docker containers.	DEPLOY UN-DEPLOY	ICoViCS RUI
5	urn:uuid:setower:msaas:seDeployment:FearlessMariner_001	Fearless Mariner 001	Deployment for the Fearless Mariner composition. Uses Terraform and Puppet scrips for deployment purposes.	DEPLOY UN-DEPLOY	RTI Web View ICoViCS RUI

Figure 3-30: AIMS Deployment Tool.

The deployment process uses the following tools:

- Terraform** to create the Infrastructure.
- Puppet** to install and configure the applications.
- Docker Compose** to launch and run the required combination of Docker containers.

These tools support both AWS and VMware (used by UKCloud). Figure 3-31 summarises the deployment process using ‘Infrastructure as Code’ approaches.

3.5.5.3.3 Test #3 Outcome/Analysis

The objective was to determine how ‘Service Deployments’ can be stored and discovered.

This experiment has:

- Demonstrated the ability to define a simulation ‘stack’ as code.
- Demonstrated the ability to create (and re-create) a simulation deployment from scratch.

There needs to be a suite of deployment automation tools sitting between the registry (for defining deployments) and the infrastructure providers (for delivering deployments) in order to succeed in automating (or semi-automating) the deployment of simulations.

The prototype deployment tool, along with the creation Infrastructure as code, is good step towards semi-automating the deployment of simulations.

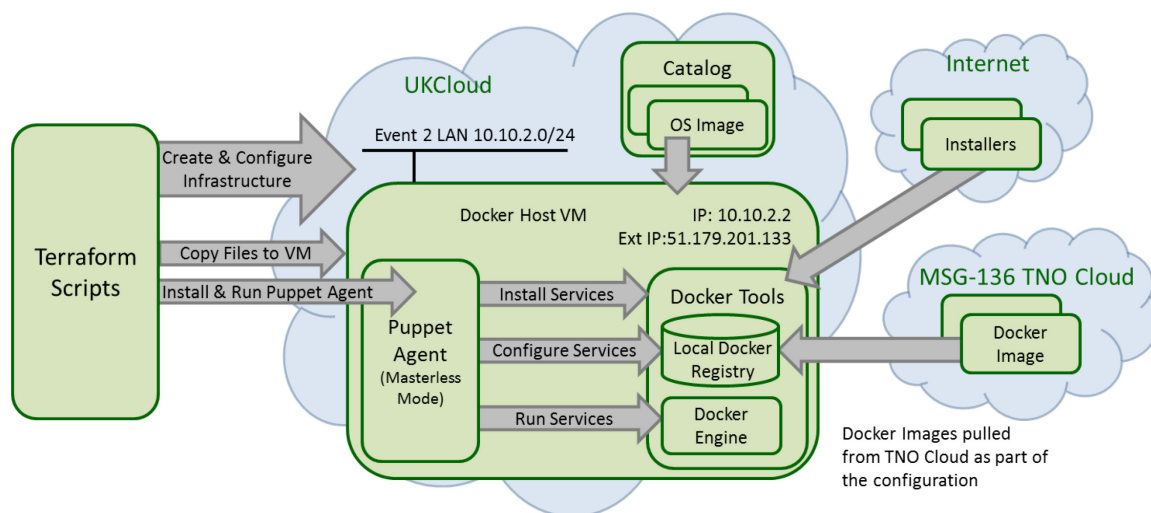


Figure 3-31: Infrastructure as Code Deployment.

3.5.5.4 Test #4

Determine how to deploy the same simulation deployment for concurrent multiple events.

3.5.5.4.1 Test #4 Experimental Setup

This test case will develop a prototype simulation deployment tool for deploying concurrently running deployments, as shown in Figure 3-32.

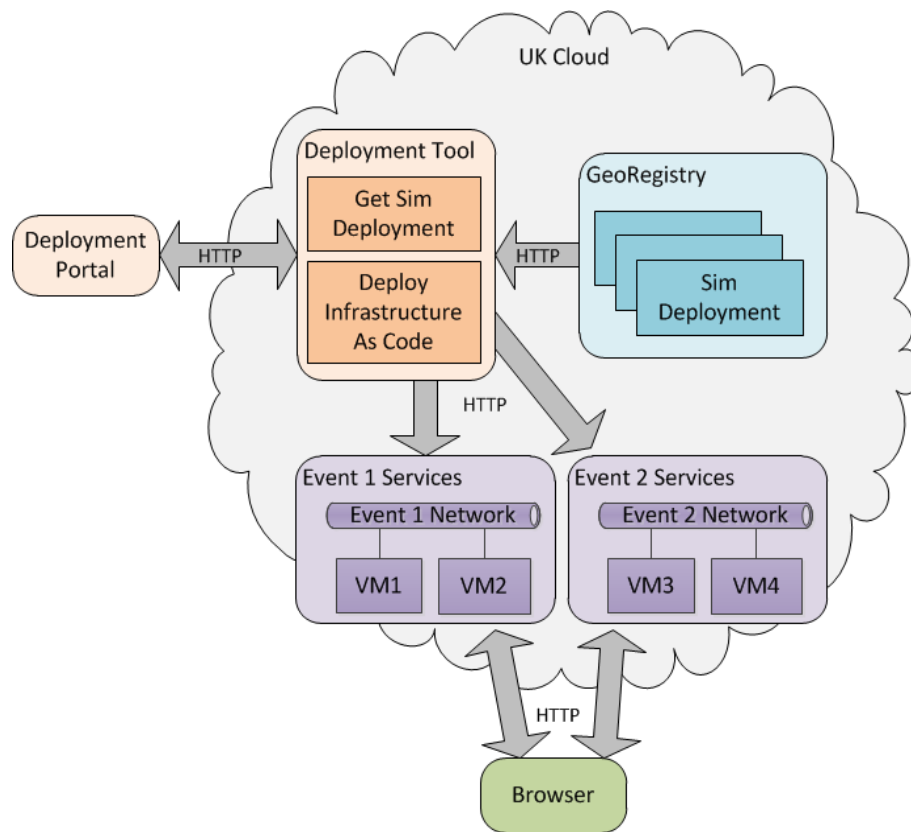


Figure 3-32: Concurrent Event Setup.

3.5.5.4.1.2 Concurrent Events

The aim of this task is to deploy the same Federation multiple times in the same cloud.

Infrastructure as Code templates created in test #3 were reused to demonstrate simultaneous execution of Simulation Deployments.

Each event ran concurrently but separated from each other. Access to the Simulation services for each event was via a browser and/or local site tools, e.g., Google Earth.

3.5.5.4.2 Test #4 Observations

Each simulation event runs concurrently but separated from each other at the network level. The following tools are used for the deployment of each event. The scripts are tailored for each event configuration:

- Terraform** to create the Infrastructure.
- Puppet** to install and configure the applications.
- Docker Compose** to launch and run the required combination of Docker containers.

Access to the Simulation services for each event is via a browser (Pitch Web GUI) and via Google Earth, as shown in Figure 3-33.

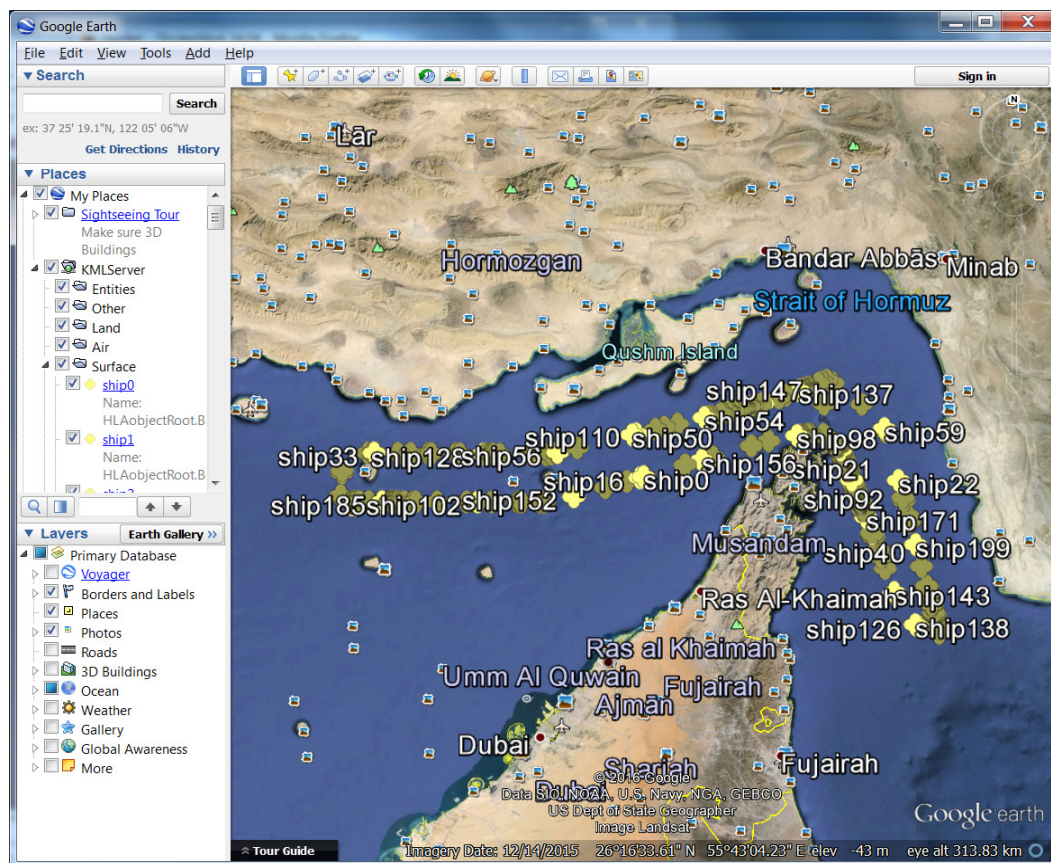


Figure 3-33: View of Event 1 Simulation.

3.5.5.4.3 Test #4 Outcome/Analysis

The objective was to determine how to deploy the same simulation deployment for concurrent multiple events.

This experiment has:

- Demonstrated the ability to run multiple con-current events.
- Demonstrated the ability to monitor key parameters across cloud services remotely.

Dynamically creating multiple con-current events is possible. The prototype deployment tool, along with the deployment process using Infrastructure as code, is good step towards semi-automating the deployment of simulations.

3.6 TEST CASE 5: CONTAINER ORCHESTRATION ENVIRONMENTS (NLD)

3.6.1 Objective/Topic/Question

Container orchestration covers generally automated configuration, coordination, and management of services. There are actually quite a number of container orchestration environments. Popular ones are Docker Native and Kubernetes.

As applications grow bigger, it is actually impossible to do without a proper container orchestration environment.

The objective of this test case is to test if Docker simulation images¹ can be used unchanged in both the Docker Native and Kubernetes orchestration environments.

This test case the described extensively in Ref. [3].

3.6.2 Assumptions/Preconditions/Boundary Conditions

None.

3.6.3 Systems and Interfaces

The following Docker images in the MSG-136 MSaaS Registry are used in this test case:

- Pitch CRC;
- Pitch WebGUI;
- Damage Server;
- KML Server;
- MaK VR Forces;
- Pacer;
- Sensor Server;
- Shipsim; and
- Start.

3.6.4 Test Setup

The test setup for Docker Native and Kubernetes is:

- Two small 5-node clusters;
- Images are pulled from private 136 Docker Registry or Docker Hub;
- Compositions/Charts are obtained from a GIT repository (here GitHub); and
- Access to services in cluster is via username/password protected http/https endpoints.

Figure 3-34 illustrates the test setup. One setup is for Docker Native, another setup for Kubernetes.

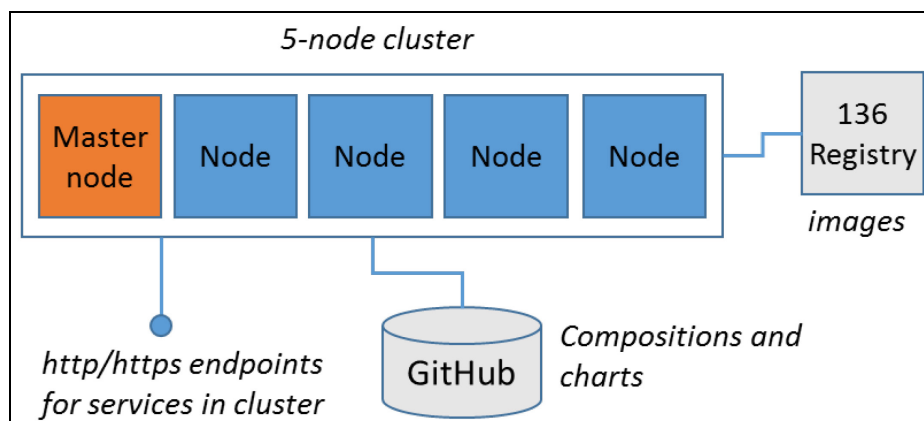


Figure 3-34: Test Setup.

¹ Based on the Docker LRC base images.

Figure 3-35 shows the Docker Native and Figure 3-36 the Kubernetes test setup in detail.

- **Docker Native:**

- Master Node 01 runs management components (all containers).
- Weave overlay network between hosts 02 – 05.
- User/Password access to services via Nginx reverse proxy.
- Docker compose files are obtained from GitHub and can be started via the Compose UI.

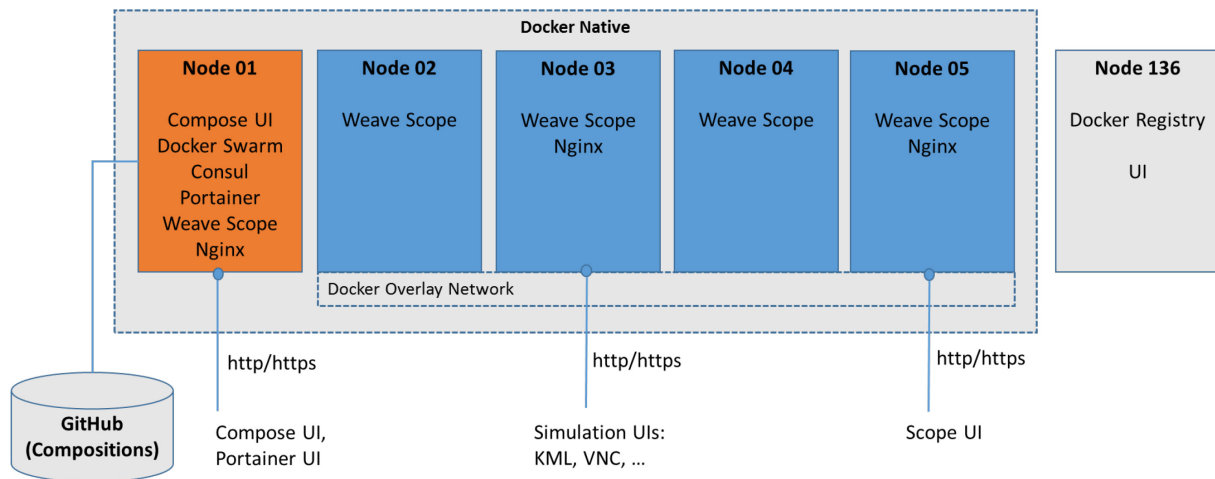


Figure 3-35: Docker Native Setup.

- **Kubernetes:**

- Master Node 01 runs management components (all containers).
- Weave overlay network between hosts 01 – 05.
- User/Password access to services via Nginx reverse proxy.
- Kubernetes chart files are obtained from GitHub and can be started via the ElasticKube UI.

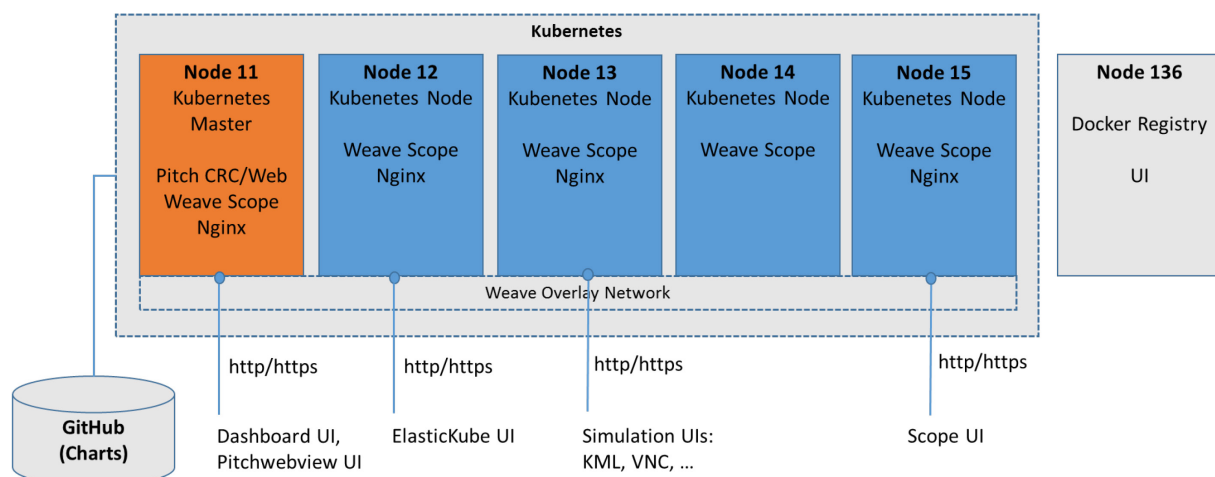


Figure 3-36: Kubernetes Setup.

3.6.5 Processes and Activities

The use case for the text case is:

- **The Modeler:**
 - Logs into the GIT repository.
 - Creates Docker composition or Kubernetes chart for a simulation (see Figure 3-37, Figure 3-38, and Figure 3-39).
 - Commits the composition/chart to the GIT repository.
- **The Simulation Controller:**
 - Logs into orchestration environment.
 - Navigates to the available compositions/charts.
 - Initiates a simulation execution by clicking on a composition/chart in the Compose UI (see Figure 3-40) or ElasticKube UI (see Figure 3-41).
 - The containers are pulled on demand from the Docker Registry and deployed in the cloud.
 - Monitors the simulation execution.
 - Terminates the simulation execution.

3.6.6 Observations

Some technical observations:

- Kubernetes does not support user defined MAC addresses for container network interfaces; work around by running the Pitch CRC and VR-Forces on host 11 on the same Weave network, but 'outside' Kubernetes. The CRC is added as external service to the Kubernetes DNS and the Kubernetes pods will find the CRC in the usual way via its DNS entry.
- Multicast between Kubernetes pods is not enabled by default in the Weave overlay network.
- In Docker Native containers with external facing interfaces must be scheduled on specific hosts for access through Nginx (e.g., KML Server). In Kubernetes these constraints do not matter and containers can be scheduled on any host.

```
1  version: '2'
2
3  services:
4    xserver:
5      image: app-docker136.hex.tno.nl:443/library/xserver
6      ports:
7        - "30000:8080"
8      environment:
9        - constraint:node==app-docker03.hex.tno.nl
10     networks:
11       - x11
12
13  networks:
14    x11:
15      driver: overlay
```

Figure 3-37: Docker Native X-Server Composition.

```
1  apiVersion: v1
2  kind: ReplicationController
3  metadata:
4    name: xserver-rc
5    labels:
6      heritage: helm
7  spec:
8    replicas: 1
9    selector:
10     app: xserver-app
11   template:
12     metadata:
13       labels:
14         app: xserver-app
15     spec:
16       containers:
17       - name: xserver
18         image: app-docker136.hex.tno.nl:443/library/xserver
19         ports:
20         - containerPort: 8080
21           protocol: TCP
22           name: http
23         - containerPort: 6000
24           protocol: TCP
25           name: x11
26         imagePullPolicy: Always
```

Figure 3-38: Kubernetes X-Server Replication Controller Specification.

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: xserver-sv
5    labels:
6      heritage: helm
7  spec:
8    selector:
9      app: xserver-app
10   type: NodePort
11   ports:
12   - name: http
13     port: 8080
14     targetPort: 8080
15     nodePort: 30000
16     protocol: TCP
17   - name: x11
18     port: 6000
19     targetPort: 6000
20     nodePort: 31000
21     protocol: TCP
```

Figure 3-39: Kubernetes X-Server Service Specification.

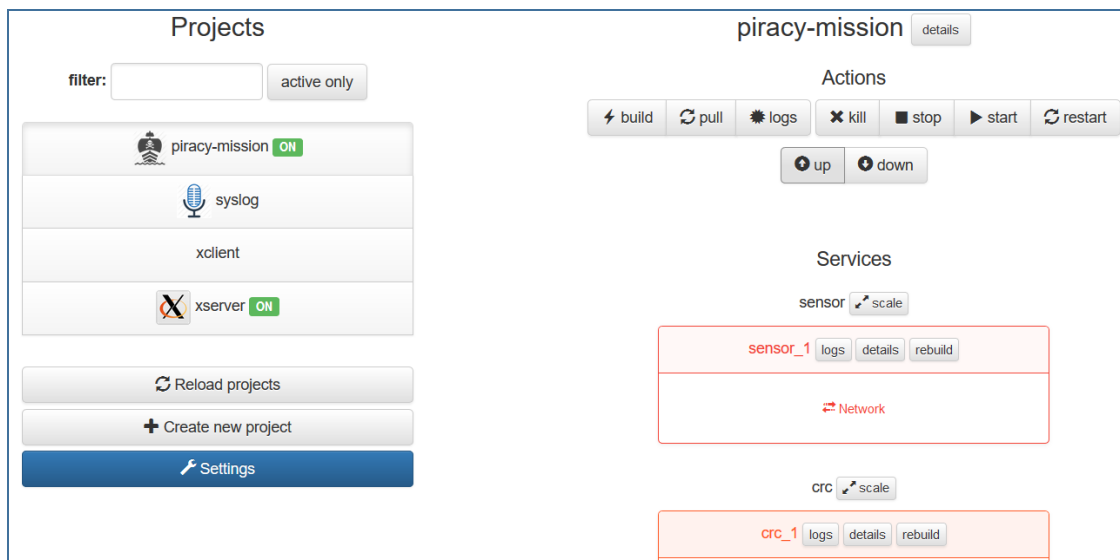


Figure 3-40: Compose UI.

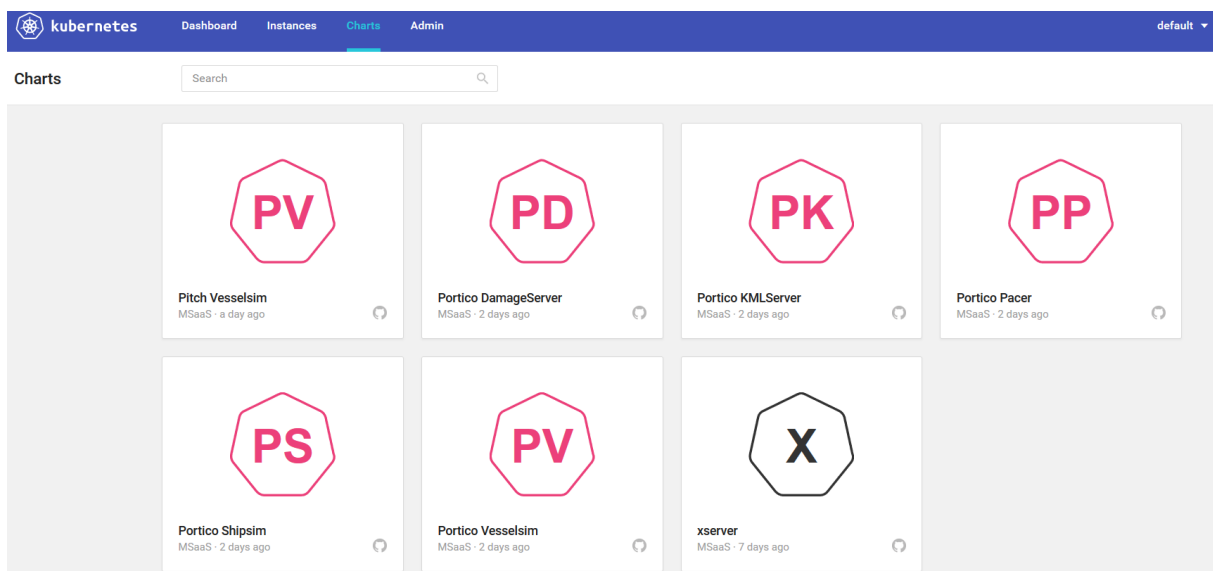


Figure 3-41: ElasticKube UI.

3.6.7 Outcome/Analysis

The (Simulation) containers can run in either environment *unchanged*.

Both Pitch (TCP) and Portico (Multicast) RTIs can be used in either environment.

However:

- Kubernetes requires more verbose configuration to define a composition.
- Differences in dealing with data volumes.
- Differences in networking models.

- Important for performance is the overlay network (e.g., Docker default, Flannel, Weave, etc.).
- Some technical caveats (e.g., use of proxies, overlay network configuration, DNS issues).
- Docker Native Orchestration easiest to set up.
- But Kubernetes offers more functionality and more concepts to use (too much to discuss here).

Chapter 4 – SIMULATION SERVICES EXPERIMENTS

4.1 OVERVIEW

In this strand of experiments, the aim is to collect a set of available services provided by the members and test their interfaces accessing them as a potential user/consumer. This activity should then produce valuable feedback both for architecture and for service instructions and definitions.

A first set of services has been identified as follows and can be viewed as a list of ABBs:

- Synthetic Environment Service (i.e., GER SES);
- Route Planning Service (i.e., NOR RPS);
- Cloud Orchestration Solution (i.e., MSCOE – Leonardo ITA OCEAN); and
- Computer Generated Forces Service and C2-Sim gatewaying Service (i.e., ITA SGA and Gateway).

These services are connected via VPN over the Internet and a series of point-to-point/service-to-service tests is performed: if a common cloud environment for testing is available, it is used as a hub for these tests.

Depending on each service owner's decision, the services can be deployed directly into the cloud provider's infrastructure or made available over the Internet through it.

4.2 TEST CASE 1: RPS, SES, AND SIMSYS (DEU, NOR)

4.2.1 Objective/Topic/Question

“Is it possible to build a simulation environment that connects RPS, SES and a SimSys (according to MSG-136 MSaaS approach)?”

4.2.2 Assumptions/Preconditions/Boundary Conditions

- Working cloud environment.
- Services/SimSys available (esp. security/access issues solved).

4.2.3 Systems and Interfaces

- NOR RPS.
- DEU SES.
- DEU SimSys 1 (PAXSEM).
- VPN/SOAP/WFS/http.

4.2.4 Test Setup

The test setup is shown in Figure 4-1.

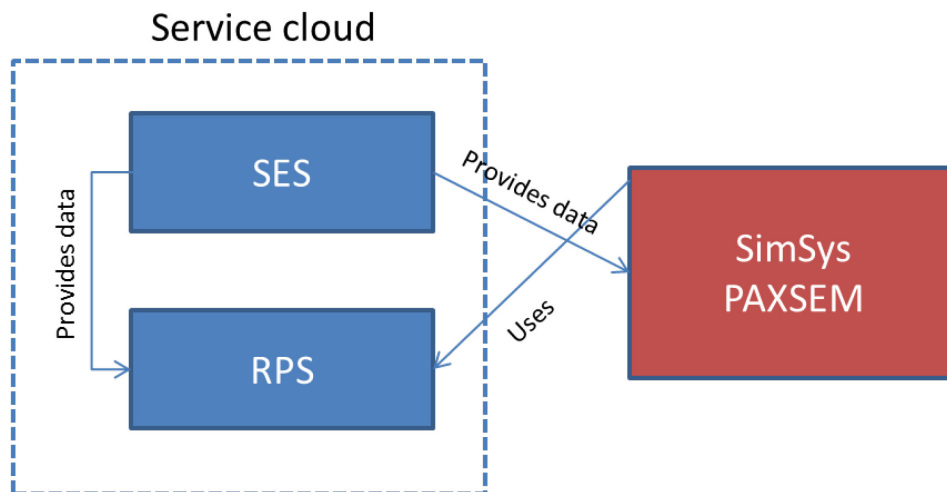


Figure 4-1: Experimental Setup.

4.2.5 Processes and Activities

Scenario:

- The scenario is located in Meppen, Germany – the terrain data for the SimSys PAXSEM and RPS is provided by the SES.
- A squad team of 2 blue tanks simulated by PAXSEM has to move to a destination taking into account known threats.
- The route is computed using the RPS service which is then used by the squad team.
- During the movement, the squad team detects other hostile units which force the squad team to replan their route using the RPS.

Tests:

- Test#1: Cloud environment (SES, RPS) online and available.
- Test#2: RPS can connect to SES.
- Test#3: RPS can fetch data from SES.
- Test#4: SimSys can connect to RPS.
- Test#5: SimSys can call RPS and get route from RPS.

4.2.6 Observations

- Errors during service calls.
- Service calls sometimes interrupted. Manual call necessary.

4.2.7 Outcome/Analysis

- Yes, a simulation environment that connects RPS, SES and a SimSys is possible.
- Infrastructure has to be improved to run stable.
- Service calls have to be implemented exactly according to description.

4.3 TEST CASE 2: INDIVIDUAL AND TEAM TRAINING FOR NAVAL C4ISTAR OPERATION (NATO-MSCOE)

4.3.1 Objective/Topic/Question

Individual, Collective (Team) Combat Management System (CMS) operators task trainer.

4.3.2 Assumptions/Preconditions/Boundary Conditions

None.

4.3.3 Systems and Interfaces

A simulation user can instantiate a training session in seconds and access to the web based Equipment Simulators (i.e., navigation system, radars and other sensors) directly from the OCEAN MS-IaaS/PaaS Orchestrator interface.

The following services/images (provided by Leonardo) are used: CMS Naval Simulator, Sensors and Equipment Simulators (NAVS, RAN21S, LRAS, OAS) using DIS/HLA, SGA (Scenario Generator and Animator).

The simulation suite reproduces main onboard ship C4ISTAR equipment in order to stimulate ship's combat management system (simulated or real).

4.3.4 Test Setup

The test setup for Naval C4ISTAR Operation session, as shown in Figure 4-2, is:

- Discovery of needed resources for C4ISTAR operational;
- Compose a training session based on discovered resources; and
- Execute the session in seconds/minutes.

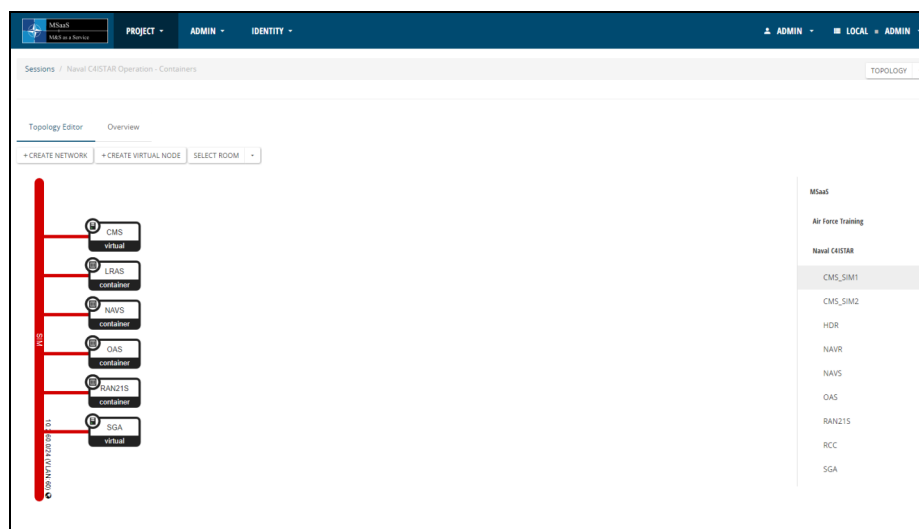


Figure 4-2: Naval C4ISTAR Session Setup.

4.3.5 Processes and Activities

- Discovery/Composition and Execution with OCEAN MS-IaaS/PaaS Orchestrator:
 - Logs into the web interface of the orchestrator;
 - Creates a new session;
 - Create DIS simulation network;
 - Add sensors containers from the dropdown menu;
 - Add CMS and SGA vms from the dropdown menu;
 - Connect container and virtual nodes to the simulation network; and
 - Press start session to build the overall virtual environment.
- CMS activities:
 - Right click on the generated nodes and open the web interface of each service; and
 - Use the interfaces for C4ISTAR operations, as shown in Figure 4-3.

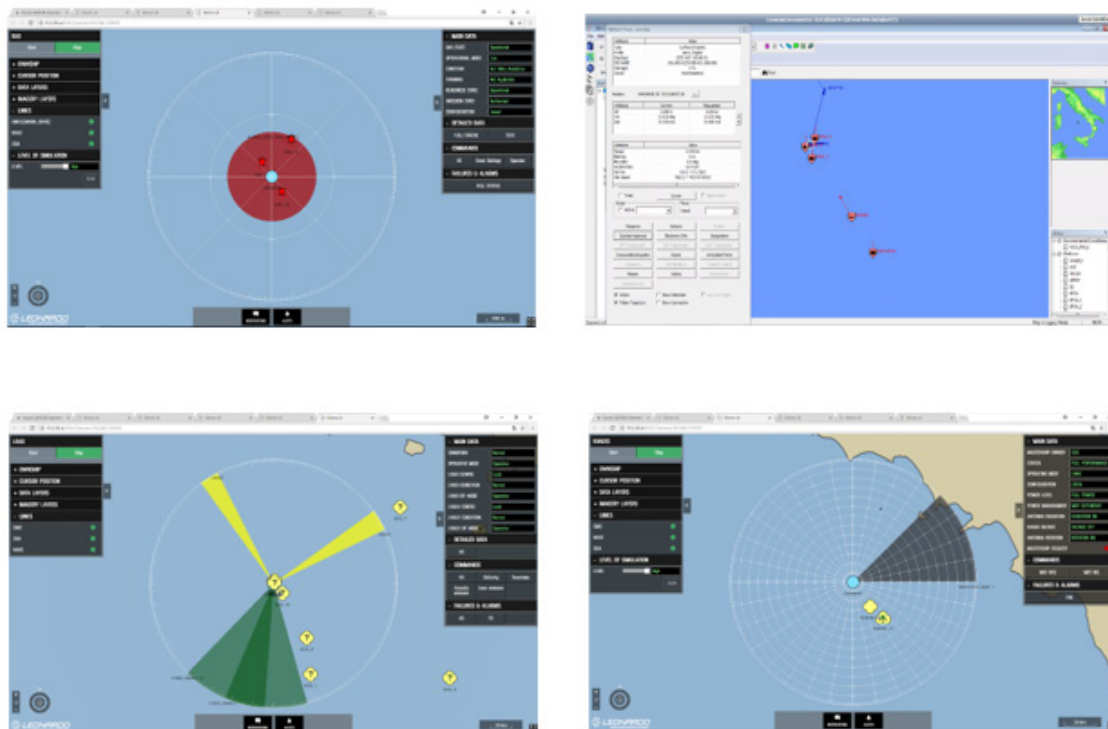


Figure 4-3: Naval C4ISTAR Operations.

4.3.6 Observations

Some technical observations:

- Mixing virtual machines and containers is possible.
- Virtual SDN networks (VXLAN) are shared between virtual and containers nodes.
- In this demo VMware Integrated Containers subsystem was used, but in the future Kubernetes will be used.

4.3.7 Outcome/Analysis

OCEAN is a third party (Leonardo Company) software build on top of OpenStack for rapid and agile orchestration.

Simulation users can execute training and exercise without any additional support personnel whenever needed.

Multi-tenancy (more users can perform the same training session at the same time)

4.4 TEST CASE 3: INDIVIDUAL TRAINING OF RADIO/C2 OPERATOR (NATO-MSCOE)

4.4.1 Objective/Topic/Question

Train (Task Trainer) how to operate radio and C2 system.

4.4.2 Assumptions/Preconditions/Boundary Conditions

None.

4.4.3 Systems and Interfaces

A radio operator (Simulation User) accesses a MS-SaaS Virtual Immersive application through the OCEAN MS-IaaS/PaaS Orchestrator.

The following services (provided by Leonardo) are used: Morpheus (Synthetic/Virtual Immersive Environment), SVC (Radio Communication Networking Service), SGA (Scenario Generator and Animator)

4.4.4 Test Setup

The test setup for RADIO/C2 Operator session, as shown in Figure 4-4, is:

- Discovery of needed resources for RADIO/C2 Operator session;
- Compose a training session based on discovered resources; and
- Execute the session in seconds/minutes.

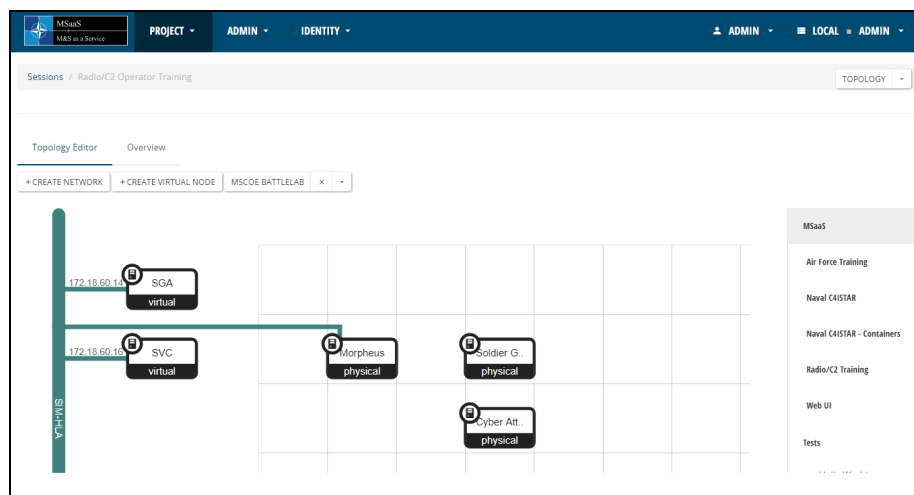


Figure 4-4: Radio/C2 Operations.

4.4.5 Processes and Activities

- Discovery/Composition and Execution with OCEAN MS-IaaS/PaaS Orchestrator:
 - Logs into the orchestrator Web Interface;
 - Creates a new Session;
 - Create HLA Simulation Network;
 - Connect Morpheus to HLA network;
 - Add SVC and SGA virtual nodes from the dropdown menu;
 - Connect virtual nodes to the simulation network; and
 - Start session to build the overall Virtual/Physical Environment.
- CMS activities:
 - Right click on the generated nodes and open the RDP Web Interface of each node.
 - Start SVC and SGA to stimulate Morpheus Virtual Environment.
 - The Scenario to be simulated is composed by soldiers moving on a land map and each one equipped with a Radio device.
 - The user is immersed into a virtual environment (Morpheus) representing a military wagon (i.e., Lince, VBM 4 x 4) located at a given point on the map.
 - The wagon is equipped with a radio and a C2 panel in order to track the movements of soldiers.
 - On the C2 panel a map will be displayed with a real time tracking of soldiers.
 - As shown in Figure 4-5, the Simulation user (Morpheus) can interact with the radio equipment (SVC) and the C2 panel in order to: switch on/off the radio; increase/reduce the power; change the frequency; manage the map (i.e., zoom-in/zoom-out).



Figure 4-5: Radio/C2 Morpheus Operations.

4.4.6 Observations

Some technical observations:

- Mixing Virtual Machines and Physical nodes is possible.
- Physical VLAN networks are shared between virtual and physical nodes.

4.4.7 Outcome/Analysis

OCEAN is a third party (Leonardo Company) software build on top of OpenStack for rapid and agile orchestration.

Simulation users can execute training and exercise without any additional support personnel whenever needed.

Training as a Service is possible mixing also real and virtual worlds.



Chapter 5 – SUMMARY AND CONCLUSIONS

MSG-136 evaluated the MSaaS concept in two experiments with overall eight test cases and various tests associated with the test cases. The first experiment dealt with test cases to test the Reference Architecture, and the second experiment tested solutions for the Simulation Services.

The Reference Architecture approach of MSG-136 proved to be valid; the technology MSG-136 used was well manageable. Some minor problems that occurred in the experiments were mainly due to the use of new technologies (i.e., Docker) and therefore due to reduced maturity of the technical system and due to to-be-grown experience of the experimenter.

The experimentation results demonstrate that MSaaS is capable of realizing the vision that M&S products, data and processes are conveniently accessible to a large number of users whenever and wherever needed.

Next steps (e.g., experiments in a follow-on group) should:

- Conduct MSaaS experiments in a fully operational environment;
- Include commonly used cyber security technologies and methods;
- Create ‘best practices’ for developing services for an MSaaS implementation;
- Create initial MSaaS implementations for operational use that are accessible to all NATO nations; and
- Investigate how to best address legacy systems in an MSaaS environment.



Chapter 6 – REFERENCES

- [1] van den Berg, T.W., Cramp, A., and Siegel, B. “Guidelines and best practices for using Docker in support of HLA federations”, 2016-SIW-031, SISO SIW Fall 2016.
- [2] NATO STO: TR-MSG-136-Part-V, Modelling and Simulation as a Service, Volume 2: MSaaS Discovery Service and Metadata.
- [3] van den Berg, T.W. and Cramp, A., “Container orchestration environments for M&S”, 2017-SIW-006; SISO SIW Fall 2017.



Annex A – SIMULATION ENVIRONMENT AGREEMENTS

This section summarizes the simulation environment agreements used for the test cases in Chapter 3.

A.1 METADATA

Information about the federation agreements document itself.

A.1.1 Identification

This section describes the simulation environment agreements for Experiment A. This section follows the structure of the SISO Federation Agreements Template (FEAT).

Only the FEAT elements that are applicable to the test cases of Experiment A are addressed in this section.

A.2 DESIGN

Agreements about the basic purpose and design of the federation.

A.2.1 Scenario

A.2.1.1 Exercise Background

Combined Task Force 152 (CTF-152), working with various ships and headquarters staff from across the Gulf Cooperation Council (GCC), recently conducted a dedicated, coordinated maritime security operation in the Gulf (see Figure A-1). These types of operations are designed to track, counter and build an understanding of any illicit activity in the Gulf in order to ensure stability in these important waters.

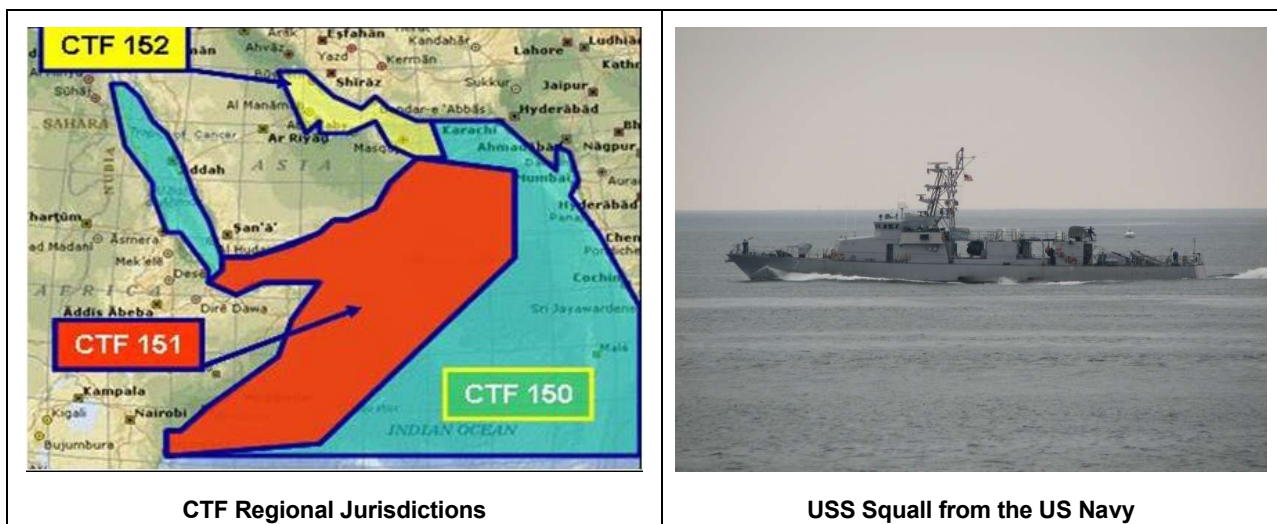


Figure A-1: Exercise Background.

The Saudi-led CTF-152, based in Bahrain, as shown in Figure A-2, used naval and air assets from the United States, United Kingdom, New Zealand and Saudi Arabia to conduct joint operations in support of Combined Maritime Forces' (CMF) mission of deterring criminal and terrorist activity in the Gulf and maintaining regional maritime security.

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

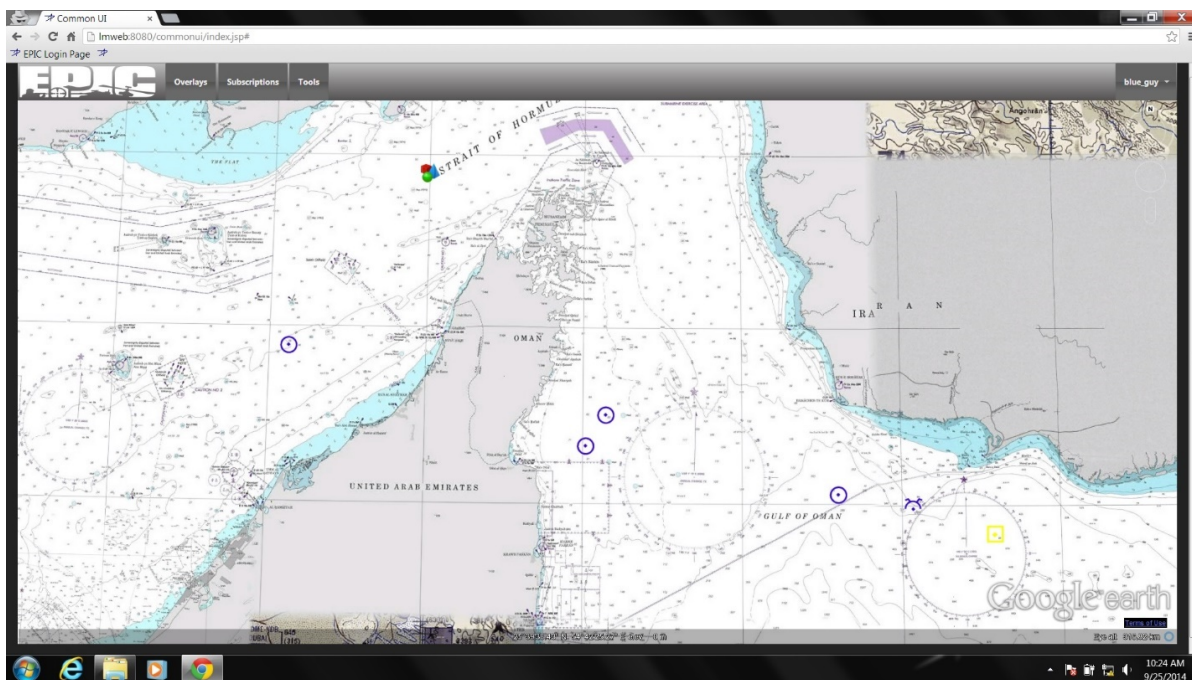


Figure A-2: Exercise Area: Strait of Hormuz and Gulf of Oman.











A.2.1.2 Situation

Friendly surface combatants are anchored in the Arabian Gulf within UAE territorial waters, off the shore of Ras Al Khaimah. These consist of a carrier and frigate, as shown in Figure A-3. The Commander's intent is to protect commercial shipping from pirates and acts of terrorism within the territorial waters of Kuwait, Bahrain, Qatar, the United Arab Emirates, Oman and Saudi Arabia. See Table A-1 for the exercise entity list.



Figure A-3: Situation Overview.

Table A-1: Exercise Entity List.

Entity	Alliance	Type	App6 Symbol
CARRIER_1	Friendly	Aircraft Carrier	
FRIGATE_1	Friendly	Frigate	
UAV_1, UAV_2	Friendly	UAV	
MH60_1, MH60_2	Friendly	Rotary Wing	
MISSILE_1, MISSILE_2	Friendly	Missile (S to S)	
CARGOSHIP_1	Neutral	Cargo Vessel	
CIVAIR_1, CIVAIR_2, CIVAIR_3	Friendly	Fixed Wing	
FASTBOAT_1, FASTBOAT_2	Enemy	Light Vessel	
STAGINGAREA_1	Enemy	Port	
CIVPOP_1	Neutral	City Population	

A.2.1.3 Phase 1: Small Craft Threat

As shown in Figure A-4:

- 1) A small surface vessel approaches the friendly formation at a high speed from the north eastern coast line. The Frigate takes steps to identify the vessel as an imminent threat.
- 2) Following warnings, the Frigate fires upon the small craft, destroying it.

A.2.1.4 Phase 2: Threat Intelligence Report

As shown in Figure A-5:

- 1) The Commander requests vessel tracking information from the regional coastal surveillance authority which indicates that the vessel left a small island port in the Gulf of Oman.
- 2) Allied ground forces control a UAV asset to gather imagery of the location and forward intelligence to the Frigate. Another fast boat is monitored leaving the location on a similar course.
- 3) The Commander assesses additional status reports, weather conditions, and the relative position of the Frigate and the threat locations within the relevant territorial waters and countries.

A.2.1.5 Phase 3: Course of Action Analysis

As shown in Figure A-6:

- 1) The Commander prepares several Courses of Action, including a Direct Action against the fast boat and the enemy forces identified on the island staging area.
- 2) Weapon pairing to targets is performed on the combat system, including an assessment of the indirect firing lines relative to the terrain.

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

- 3) Airspace de-confliction is undertaken with government authorities and regional air traffic control. Both governments provide specific ROEs for firing a surface launched weapon over UAE waters and an Omani civilian populated area (Khasab).
- 4) A solution is available to engage both threats within a short window of opportunity.

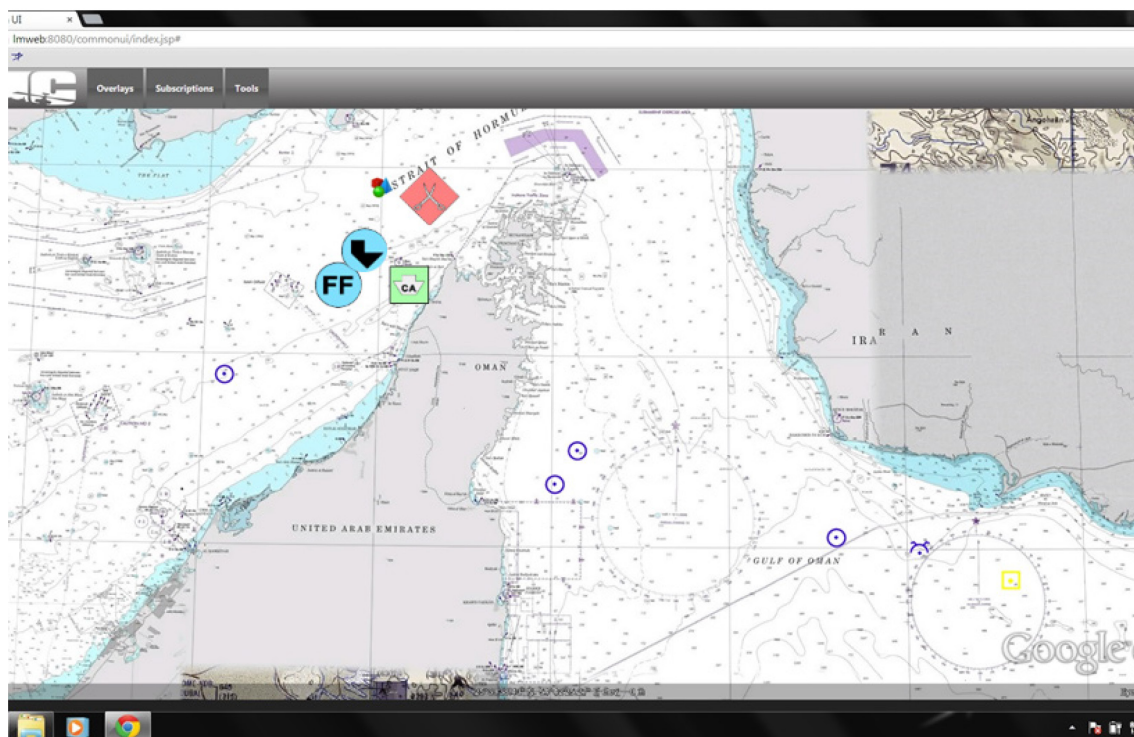


Figure A-4: Phase 1 Situation.

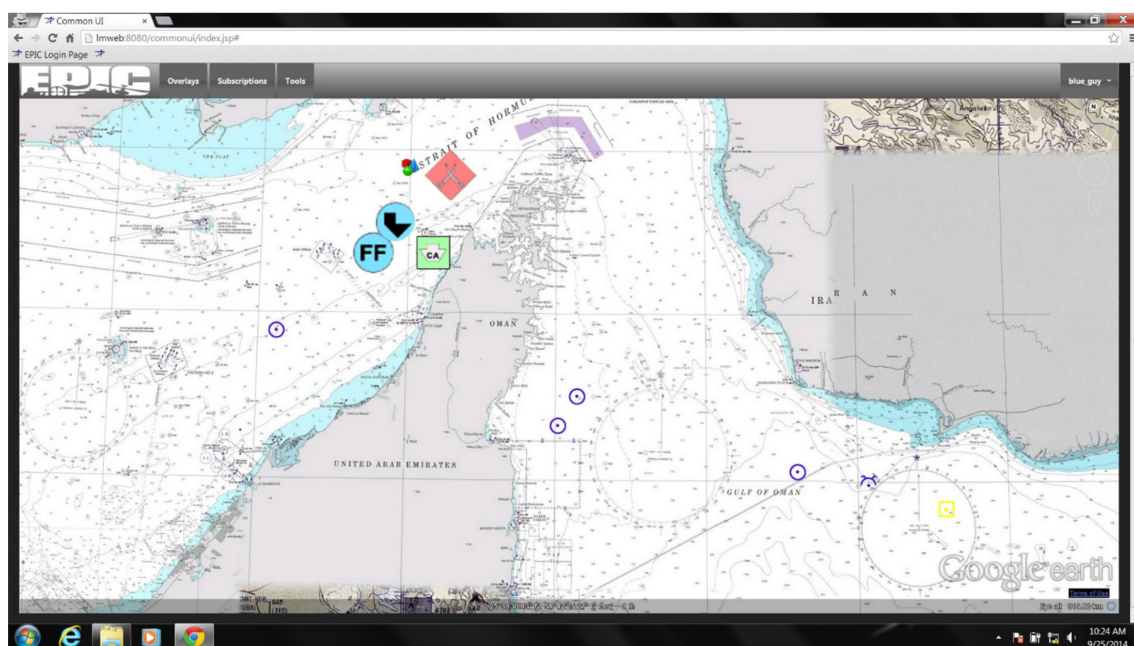


Figure A-5: Phase 2 Situation.



Figure A-6: Phase 3 Situation.

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

A.2.1.6 Phase 4: Direct Action Against Threats

As shown in Figure A-7:

- 1) Air Task Orders are issued to air assets to position to provide real time Fire Control and Battle Damage Assessment.
- 2) The Commander gives orders to the Combat Officer to engage the designated threats in accordance with the planned mission time.
- 3) The Commander maneuvers the Frigate to the required launch point.
- 4) Friendly air assets enter an observation orbit/point at a safe distance from the target area.
- 5) The Commander confirms de-confliction, and the surface to surface weapons are launched from the Frigate. The missiles impact the targets.
- 6) Friendly air assets report Battle Damage Assessment to the Commander.

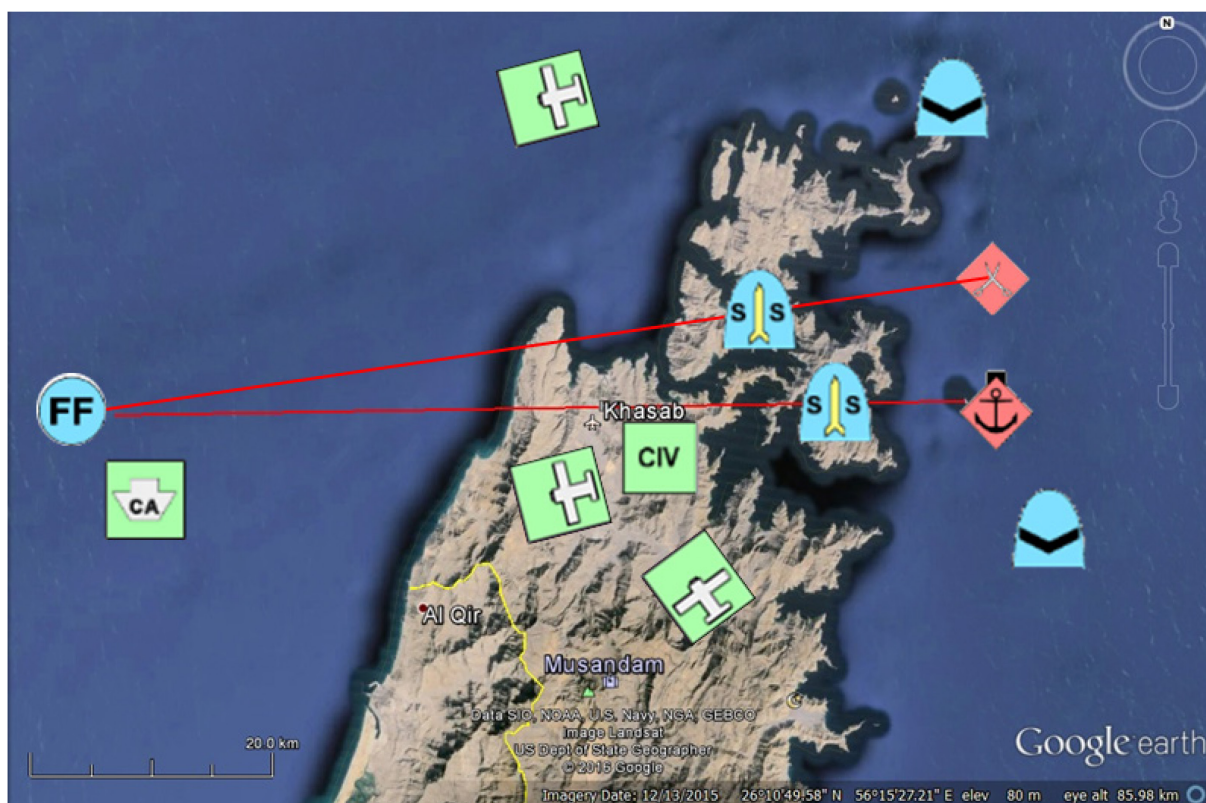


Figure A-7: Phase 4 Situation.

A.2.1.7 Phase 5: SOF Secure Target Area

As shown in Figure A-8:

- 1) The Commander informs allied Special Forces they are clear to enter the airspace and engage on the ground.
- 2) Allied SOF units fly rotary wing platforms onto the island to secure the area.



Figure A-8: Phase 5 Situation.

A.2.2 Conceptual Model

An event-service matrix (see Table A-2) is used to describe the tasks that would need to be performed by exercise participants, and to describe the events that may occur during an exercise.

The event-service matrix is in fact a tabular representation of an activity model, showing the tasks, events and some of the main information items over time.

The event-service is not complete and has been used in the engineering process walkthrough to identify potential services. The identification of potential services is an activity in DSEEP step 3.

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

Table A-2: List of Events.

SERIAL ID	EVENT #	EVENT/TASK DESCRIPTION	PARTICIPANT/S	ACTIVITIES	SERVICE/S
0	0	Populate Registry	Registry Administrator	Create 10 dummy simulation compositions in the registry	Repository service
0	0	Manage Registry	Registry Administrator		
0	0	Maintain Registry	Registry Administrator		
0	0	Create Repository	Repository Administrator		
0	0	Populate Repository	Repository Administrator		
0	0	Maintain Repository	Repository Administrator		
0	0	Create Service	Service Provider		
0	0	Validation Test Service	Service Provider		
0	0	Supply Service	Service Provider		
0	0	Initiate event planning and define requirements			
0	0	Create new event	Exercise Controller	User logs in to MSaaS Portal and creates new event (name, date, etc.)	Event planning service/ WORKFLOW
0	0	Assign roles	Exercise Controller	User assigns roles to persons (like event director, support staff, etc.)	
0	0	Specify event objectives	Exercise Controller	User specifies event objectives etc.	
0	0	Specify operational scenario	Exercise Controller	User specifies operational scenario (selects an existing scenario from the scenario library)	
0	0	Specify event partners	Exercise Controller	User specifies event partners (e.g., other nations, organizations)	

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

SERIAL ID	EVENT #	EVENT/TASK DESCRIPTION	PARTICIPANT/S	ACTIVITIES	SERVICE/S
0	0	Define simulation requirements and discover services	Sim Controller	0	
0	0	Specify simulation requirements	Sim Controller	User specifies simulation requirements to achieve event objectives	WORKFLOW
0	0	Identify networks and connections	Sim Controller	User identifies required networks and connections (e.g., national networks, coalition networks)	WORKFLOW
0	0	Identify available services	Sim Controller	User searches MSaaS Portal for available simulation services (local, remote)	Repository service
0	0	Select required services	Sim Controller	User selects most appropriate services	WORKFLOW
100	0,1	Exercise Orchestration (Develop simulation environment)	0	0	
0	0	Configure services	0	User configures services (ORBAT, etc.)	Composition service/ WORKFLOW
110	0,11	Initialise CGF (load terrain, create entities, create weapons, create routes, set behaviors)	Sim Controller	Sim Controller configures and launches CGF	Composition service/ WORKFLOW
120	0,12	Initialise EXCON software interface	Sim Controller	Sim Controller launches EXCON	Composition service/ WORKFLOW
130	0,13	Initialise event logging	Sim Controller	Sim Controller launches Logging	Logging service
140	0,14	Initialise WebCOP (exercise SA for players)	Sim Controller	Sim Controller configures and launches WebCOP	COP SERVICE
150	0,15	Initialise player controls	Sim Controller	Sim Controller launches player controls (Sense target, select target, sound horn, fire weapon, issue orders, request report)	Composition service/ WORKFLOW

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

SERIAL ID	EVENT #	EVENT/TASK DESCRIPTION	PARTICIPANT/S	ACTIVITIES	SERVICE/S
151	0,151	Configure sensors	Sim Controller	Sim Controller configures and launches sensor/weapon simulators	Sensor service, Weapon Service
200	0,2	Create all surface vessel tracks	Sim Controller	CGF moves all vessels according to routes	Traffic Generation service
310	0,31	Create COA mission tactical graphics in KML	Sim Controller	Push KML graphics into WebCOP/CGF	COP service, Viewing Service, Icon Service
333	0,333	Prepare Rules Of Engagement from HICON	Sim Controller	Event triggered by target selection on WebCOP	COP service
0	0	Verify simulation environment	Sim Controller, Exercise Controller	User verifies that the simulation environment satisfies the event requirements	WORKFLOW
0	0	Store and Publish the simulation composition	Registry Administrator, Sim Controller	User saves the simulation composition and publishes it the Registry	REGISTRY SERVICE
0	0	Service Discovery	Sim Controller, Exercise Controller, Participant	Find updated services	Repository service
0	0,01	Search for a training scenario	Participant	User searches for an individual training exercise	SEARCH SERVICE/WORKFLOW
0	0	Select training scenario	Participant	User selects the individual training exercise	WORKFLOW
0	0	Send training event request	Sim Controller, Exercise Controller, Participant	User sends request to run training exercise (services and humans are notified)	WORKFLOW
0					
1000	1	Execute Scenario	FASTBOATs, UAVs, Civil Aircraft, Civil Cargo Ships	CGF moves all entities throughout Phase 1	COP SERVICE
1100	1,1	Small surface vessel approaches the friendly formation at a high speed	FASTBOAT_1, FRIGATE_1	CGF to move vessel position iaw route and terrain	Fastboat (ship) service
1110	1,11	Detect surface vessel	FASTBOAT_1, FRIGATE_1	Sensor simulation and/or WebCOP	Sensor service

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

SERIAL ID	EVENT #	EVENT/TASK DESCRIPTION	PARTICIPANT/S	ACTIVITIES	SERVICE/S
1120	1,12	Identify surface vessel as friend or foe	FASTBOAT_1, FRIGATE_1	Vessel marked as threat on combat system/WebCOP	COP SERVICE
1200	1,2	Frigate warns fast boat	FASTBOAT_1, FRIGATE_1	Participant operates horn, CGF determines fastboat action	Frigate (ship) service
1300	1,3	Frigate fires at fast boat	FASTBOAT_1, FRIGATE_1	Participant selects weapon and ammunition, selects target	Weapon service
1310	1.3.1	Combat Officer fires weapon	FASTBOAT_1, FRIGATE_1	Participant fires weapon, weapon effects determined	Weapon service
1320	1.3.2	Fastboat under fire	FASTBOAT_1, FRIGATE_1	CGF to determine weapon effects on fastboat	Weapon Effect service
2100	2,1	Commander requests vessel tracking information	FASTBOAT_1, FRIGATE_1	Vessel track information displayed on WebCOP/Combat System	Vessel traffic service, COP service
2200	2,2	Commander requests intelligence report from allied UAV	FRIGATE_1, UAV_1, Exercise Controller/Roleplayer	Participant interacts with IntelOperations roleplayer over voicenet	WORKFLOW
2210	2.2.1	Move UAV	UAV_1	CGF moves the UAV into position relative to target area	UAV service
2220	2.2.2	Collect imagery	UAV_1	Stealthviewer captures synthetic view of Area Of Interest	UAV service
2230	2.2.3	Send imagery / intelligence report	FRIGATE_1, UAV_1, Exercise Controller/Roleplayer	IntelOperations roleplayer sends imagery/ Intel report to Commander	WORKFLOW
2300	2,3	Commander assesses additional status reports, weather conditions, and the relative position	FASTBOAT_2, FRIGATE_1, UAV_1	Commander relies on C4iSR and WebCOP for this information	COP service

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

SERIAL ID	EVENT #	EVENT/TASK DESCRIPTION	PARTICIPANT/S	ACTIVITIES	SERVICE/S
3100	3,1	Commander prepares several Courses of Action	FASTBOAT_2, FRIGATE_1, UAV_1	Commander performs/selects pre-prepared COA: Direct Action	COA SERVICE
3200	3,2	Weapon pairing to targets are performed	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1	0	Weapon Pairing service
3210	3.2.1	Weapon pairing to targets are performed on the combat system	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1	Commander selects targets on WebCOP and/or via player controls	COP service
3220	3.2.2	Assessment of the indirect firing lines relative to the terrain	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1	CGF calculates weapon routes iaw terrain	ROUTE PLANNING SERVICE
3300	3,3	Airspace de-confliction is undertaken	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1, UAV_1,2, CIVAIR_1,2,3, CARGOSHIP_1	CGF determines shoot/no shoot iaw entity positions	Airspace Deconfliction service
3310	3.3.1	Airspace de-confliction is undertaken with government authorities	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1, CARGOSHIP_1	CGF determines shoot/no shoot iaw entity positions	Airspace Deconfliction service
3320	3.3.2	Airspace de-confliction is undertaken with regional air traffic control	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1, CIVAIR_1,2,3	CGF determines shoot/no shoot iaw entity positions	Airspace Deconfliction service

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

SERIAL ID	EVENT #	EVENT/TASK DESCRIPTION	PARTICIPANT/S	ACTIVITIES	SERVICE/S
3330	3.3.3	Governments provide specific ROEs for launching air weapons in civilian airspace/territory	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1, CIVAIR_1,2,3, CIVPOP_1, Exercise Controller/Roleplayer	HICON roleplayer provides ROEs to Commander	WORKFLOW
3400	3,4	Weapon engagement solutions for both targets are provided with a specific timeframe	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1	CGF provides solution on WebCOP	COP service
4100	4,1	Air Task Orders are issued to air assets	FRIGATE_1, UAV_1, UAV_2	Commander issues orders to air assets on WebCOP and/or via player controls	UAV service
4200	4,2	Commander gives orders to engage the designated threats	FRIGATE_1	Commander selects targets on WebCOP and/or via player controls	COP Service
4300	4,3	Commander maneuvers the Frigate to the required launch point	FRIGATE_1	Commander selects launch point and orders Frigate to move to point	Frigate service
4400	4,4	Friendly air assets enter an observation orbit/point	UAV_1, UAV_2, FASTBOAT_2, STAGINGAREA_1	CGF moves friendly air assets iaw with issued ATOs	UAV service
4500	4,5	Commander confirms de-confliction and launches missiles at targets	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1	0	Weapon Service
4510	4.5.1	Commander confirms de-confliction	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1, CIVPOP_1, CIVAIR_1,2,3, CARGOSHIP_1	0	

ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

SERIAL ID	EVENT #	EVENT/TASK DESCRIPTION	PARTICIPANT/S	ACTIVITIES	SERVICE/S
4520	4.5.2	Commander launches the surface to surface weapons from the Frigate	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2, FRIGATE_1	Commander gives order to attack targets and/or uses player controls to fire weapons	Weapon Service
4530	4.5.3	The missiles impact the targets	FASTBOAT_2, STAGINGAREA_1, MISSILE_1,2	0	
4600	4,6	Friendly air assets report Battle Damage Assessment to the Commander	UAV_1, UAV_2, FASTBOAT_2, STAGINGAREA_1, FRIGATE_1	0	Damage Assessment service?
5100	5,1	Commander informs allied Special Forces they are clear to enter the airspace	FRIGATE_1, MH60_1, MH60_2	0	
5200	5,2	Allied SOF units fly rotary wing platforms onto the island to secure the area	MH60_1, MH60_2, STAGINGAREA_1	0	
0	1	Exercise Controller presents After Action Review using;	Exercise Controller, Participant	0	Recording service
0	a	Simulation logging	Exercise Controller	0	
0	b	Simulation playback	Exercise Controller	0	
0	c	Simulation entity end states and exercise measures	Exercise Controller	0	
0	d	Simulation stealth views	Exercise Controller	0	
0	e	Decision Matrix for exercise/participant evaluation	Exercise Controller	0	
0	f	Participant feedback form	Participant	0	



ANNEX A – SIMULATION ENVIRONMENT AGREEMENTS

SERIAL ID	EVENT #	EVENT/TASK DESCRIPTION	PARTICIPANT/S	ACTIVITIES	SERVICE/S
0	g	Documented execution problems	Exercise Controller		
0	h	Exercise results logging	Exercise Controller	0	
0	2	Exercise Controller stores all necessary exercise products including;	Exercise Controller		
0	a	Exercise report	Exercise Controller		
0	b	Lessons learned	Exercise Controller		
0	c	Reusable products (exercise materials such as ROEs, imagery etc.)	Exercise Controller		

A.2.3 Architecture

The simulation environment consists of a set of containerized components, distributed across various compute nodes. Two types of components are distinguished: infrastructure components and simulation components. The infrastructure components manage the simulation components, whereas the simulation components provide the required simulation functionality. The simulation components can be assembled in different configurations, called compositions.

Containerized infrastructure components include:

- MSaaS Docker Registry;
- Docker Compose UI;
- Portainer; and
- Kubernetes.

Containerized simulation components are described in Annex B

A.2.4 Services

This section details (to a certain extent) some of the simulation services identified in the architecture section.

Service interaction (NSOV-4) models are used to specify how a service interacts with external agents, and the sequence and dependencies of those interactions. The NSOV-4 models used here do not specify the sequencing of an orchestrated set of services. Their purpose is to specify the general sequence of interactions that are possible for the given services.

Several other models (not shown) may be used to detail the service agreements. The data exchange model for simulation services is described in Section A.5.

More information about each service can be found in the container image descriptions in Annex B.

A.2.4.1 Weapon Service

The Weapon Service generates Munition messages upon the receipt of a WeaponFire message. The information in the WeaponFire message determines range and flight time of the munition. A sequence of Munition messages is concluded with a MunitionDetonation message, as shown in Figure A-9.

A.2.4.2 Sensor Service

The Sensor Service generates Track messages for messages about physical entities. The sensor capabilities and performance depend on the underlying sensor model, but in general the sequence of interactions is shown in Figure A-10. The service is for one sensor.

A.2.4.3 Damage Service

The Damage Service has two options to provide damage status to a physical entity:

- Via an update of the damage state attribute of the physical entity; and
- Via a damage report to the entity.

The first option involves attribute ownership transfer once the physical entity is discovered. The Damage Service uses option 1, and defaults to option 2 in case the ownership transfer fails.

The sequence of interactions for option 2 is shown in Figure A-11.

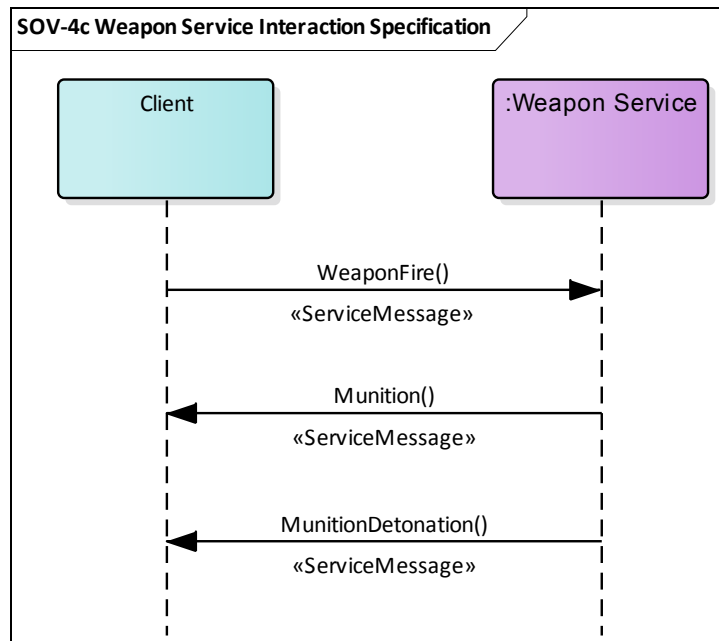


Figure A-9: Weapon Service Interactions.

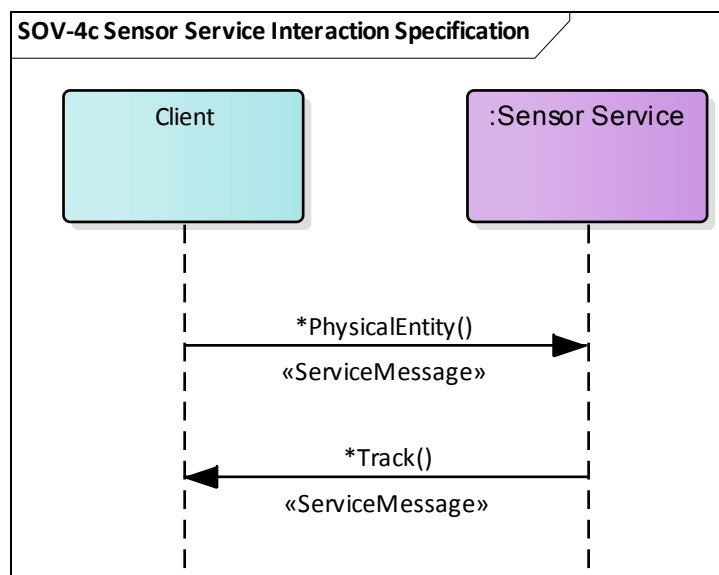


Figure A-10: Sensor Service Interactions.

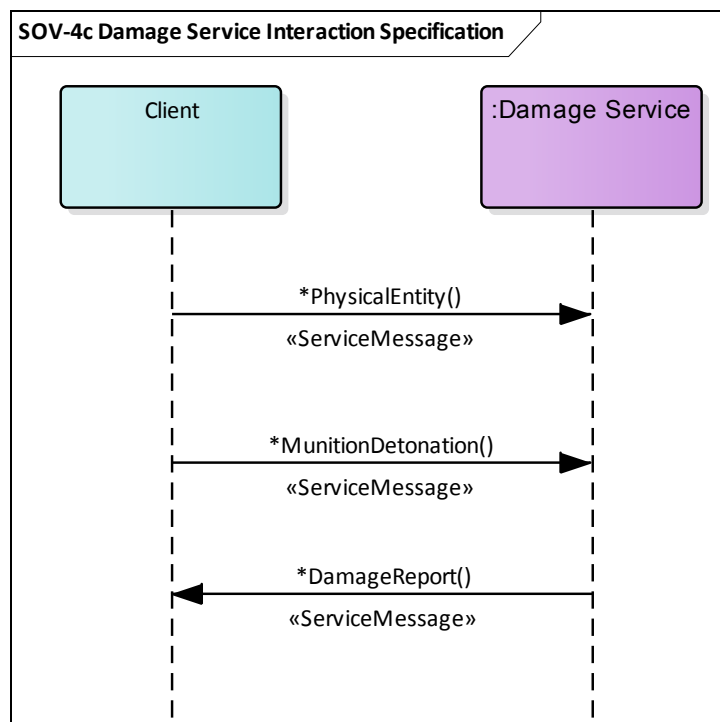


Figure A-11: Damage Service Interactions.

A.2.4.4 RWO Generation Service

The RWO Generation Service generates messages about physical entities, weapon fire and munition detonations. The sequence of interactions is shown in Figure A-12.

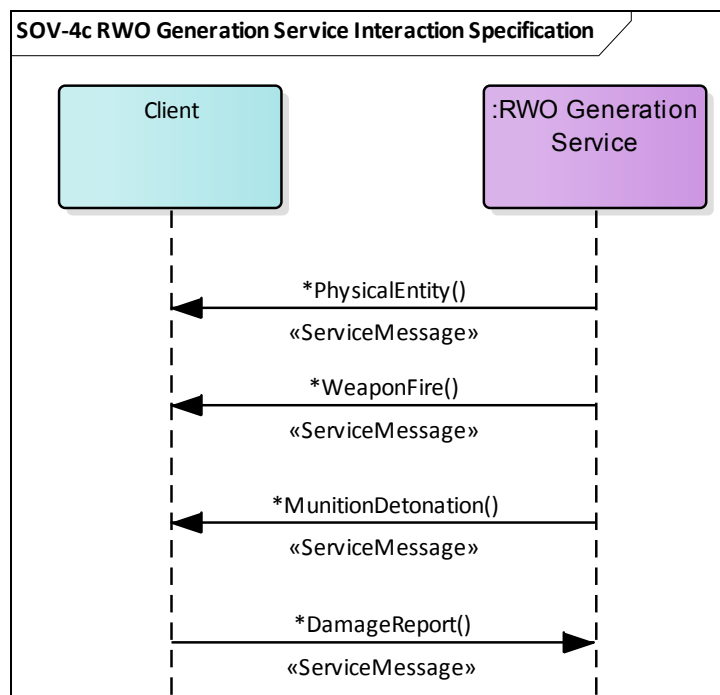


Figure A-12: RWO Service Interactions.

A.2.5 Member Applications

See container image descriptions in Annex B – Section B.2.

A.3 EXECUTION

A.3.1 Execution States

The simulation environment (consisting of containerized simulation components) has two states:

- **Bootstrap:** The central bootstrap component has not started yet. Each Local RTI component waits for central bootstrap component to start; once the bootstrap component has started, each Local RTI component will start the federate application and transitions to Execution.
- **Executing:** The central bootstrap component has started; federate applications execute.

A.3.2 Time Management

The simulation environment supports two time management modes: wall-clock time and logical time.

All federate applications support these two modes. By default, each federate application executes in wall-clock time.

Use of RPR-FOM depends on mode:

- **In time management mode:** TSO variant of RPR-FOM (the standard SISO FOM, but where all object instance attributes and interactions are marked TSO).
- **In wall-clock time:** RO variant of RPR-FOM (the standard SISO FOM).

A.3.3 Join and Resign

Federate applications may join and resign as needed (in both time management modes).

This is also required to have the ability to scale up simulation services.

A.3.4 Update Rates

As per GRIM.

A.3.5 Performance Thresholds

N/A.

A.3.6 Data Logging

Depending on the composition a recording component may be present.

A.3.7 Data Replay

N/A.

A.3.8 Monitoring

Monitoring of simulation components is provided via the infrastructure components (such as Portainer).

More detailed monitoring may be provided via the HLA LRC/CRC components.

A.3.9 Middleware Agreements

By default, the Pitch RTI is used for connecting federate applications.

Many federate applications are also provided for the Portico RTI.

A.3.10 Member Configuration

Through container command line options and environment variables. See Annex B – Section B.2.

A.4 MANAGEMENT

Container descriptions, example compositions and some of the source code for container images are maintained on GitHub.

A.5 DATA

A.5.1 Encodings

As per GRIM.

A.5.2 Data Exchange Models

- HLA RPR-FOM 2.0 and GRIM;
- HLA Damage FOM module;
- HLA Sensor FOM module; and
- HLA Empty FOM module.

A.5.3 Naming Conventions

- Default federation name: TheWorld;
- Default federate name: assigned by RTI; and
- Default object instance name: assigned by RTI.

A.5.4 Publish/Subscribe Responsibilities

These are not defined explicitly. Some of the responsibilities are captured in the description of the services in Annex B and in the service sequence diagrams in subsection A.2.4.

A.6 INFRASTRUCTURE

Simulation components are containerized.

Besides the HLA-RTI, protocols used are:

- DIS (EPIC);
- Websockets (EPIC, XServer);
- X11 (XServer); and
- HTTP/HTTPS (SES, KML Server).

A.7 MODELING

N/A.

A.8 VARIANCES

N/A.



Annex B – DOCKER CONTAINER IMAGE DESCRIPTIONS

This section contains a copy of the Docker container image descriptions that were maintained on GitHub for the test cases in Chapter 3.

B.1 GENERAL

B.1.1 Home

B.1.1.1 MSaaS Docker Registry

The Docker container images in the MSaaS Docker Registry are described in the Wiki of this GitHub project.

The following container images require license agreements:

- EPIC container images (check supplier: Lockheed Martin); and
- Pitch container images (check supplier: Pitch).

When a container image from the MSaaS Docker Registry is used in a demo (either as a foreground service or merely as a background service), please confirm use with the supplier and provide credits to the supplier for use of the container image.

Users (with sufficient access rights) can push or pull Docker images to or from the MSaaS Docker Registry. The registry also has a web UI. The addresses are as follows:

- MSaaS Docker Registry UI: <https://app-docker136.hex.tno.nl>.
- MSaaS Docker Registry for docker push/pull commands: <https://app-docker136.hex.tno.nl:443>.

For more information about accessing the registry, see MSaaS-Docker-Registry-access at <https://github.com/globalsim/msaas-A/wiki/MSaaS-Docker-Registry-access>.

To go directly to the MSaaS Docker Registry UI, navigate to <https://app-docker136.hex.tno.nl>.

B.1.1.2 Service Descriptions

Each service description has the following elements:

- Title;
- Image (registry URIs for the Docker images containing the service);
- Description;
- Synopsis;
- Docker options;
- Container options;
- Other information; and
- Examples.

B.1.2 MSaaS Docker Registry (TNO)

B.1.2.1 Obtain Access to the MSaaS Docker Registry

The first thing you need to do is arrange access to the MSaaS Docker Registry. Access to the registry is via https and is password protected. The MSaaS Docker Registry has a web UI and a Docker command line interface.

B.1.2.2 Request a User Account

Request a user account for the MSaaS Docker Registry. Send an email to tom.vandenberg@tno.nl and provide your GitHub username in the email. An account will be created for you.

B.1.2.3 Login to the MSaaS Docker Registry via the Web UI and Change Your Initial Password

Once you have an account, login to the MSaaS Docker Registry UI via a webbrowser. The URL is: <https://app-docker136.hex.tno.nl>. Change your password by selecting the Account Settings link under your user name, and click Change Password.

B.1.2.4 Login to MSaaS Docker Registry via the Docker Command Line Interface

Login and provide your username and password:

```
docker login app-docker136.hex.tno.nl:443
```

Username: <YOUR USER NAME>

Password: <YOUR PASSWORD>

B.1.2.5 Push and Pull Images

Role based access controls determine for what project repositories you can push or pull images.

- You can anonymously pull images from the public project repository:

```
docker pull app-docker136.hex.tno.nl:443/public/<IMAGE NAME>
```

- Once you are logged in you can push/pull container images to/from project repositories. For example, you can push images to a project repository <REPO NAME> for which you have the Developer role:

```
docker push app-docker136.hex.tno.nl:443/<REPO NAME>/<IMAGE NAME>
```

- If you are logged in as an NLD developer, you can for example do:

```
docker push app-docker136.hex.tno.nl:443/msaas-nld/myimagename:2.0
```

- You can pull images from a project repository <REPO NAME> for which you have the Guest role:

```
docker pull app-docker136.hex.tno.nl:443/<REPO NAME>/<IMAGE NAME>
```

- For example:

```
docker pull app-docker136.hex.tno.nl:443/library/xserver
```

B.1.2.6 Delete Images

You can delete images and repositories via the MSaaS Web Registry web UI, provided you have the proper access rights.

B.2 DOCKER IMAGE DESCRIPTIONS

B.2.1 Cesium Image (TNO)

B.2.1.1 Image

`app-docker136.hex.tno.nl:443/msaas-nld/cesium`

B.2.1.2 Description

This image is a simple (and probably inefficient) cesiumjs.org webbased viewer for KML data on a map, such as produced by the KML Server.

B.2.1.3 Synopsis

```
docker run <docker options> msaas-nld/cesium <container options>
```

B.2.1.4 Docker Options

```
-p <port number>:8080 (Required for viewing)
```

B.2.1.5 Container Options

None.

B.2.1.6 Web Address

The web address is: `http://<host>:<port>/Cesium/index.jsp`

The web address parameters are:

- `kmlserver`: one or more URLs for fetching KML data (default is <http://127.0.0.1:8090/kmlserver/entities>).
- `interval`: the interval (in seconds) for doing this or zero for fetching the data only once (default is: 5).

B.2.1.7 Example

The following example displays a map and fetches KML data from the default address:

```
http://<host>:<port>/Cesium/index.jsp
```

The following example displays a map and fetches KML data from two addresses (entities and tracks), and only once:

```
http://<host>:<port>/Cesium/index.jsp?kmlserver=http://127.0.0.1:8090/kmlserver/entities&kmlserver=http://127.0.0.1:8090/kmlserver/tracks&interval=0
```

The address 127.0.0.1 should be replaced by the host address or name of the KML Server.

B.2.2 Damage Server Image (TNO)

B.2.2.1 Image

`app-docker136.hex.tno.nl:443/msaas-nld/damageserver:pi`

`app-docker136.hex.tno.nl:443/msaas-nld/damageserver:po`

`app-docker136.hex.tno.nl:443/msaas-nld/damageserver:none`

The Pitch (pi) and Portico (po) images are based on LRC version 2. The none image (none) can be mounted as a volume to either one of the LRC base images.

B.2.2.2 Description

The Damage Server contributes to fair fight by providing entity damage reports in response to munition detonations for all federates in the federation. Damage reports are provided via an EntityDamageReport interaction for all affected entities.

The Damage Server requests ownership of each physical entity **damageState** attribute so it can update the damage state on behalf of the entity. If a federate does not divest ownership of this attribute then it is expected that the federate will accumulate damage reports for the entity and update the attribute accordingly.

Damage calculations by the Damage Server are simple and distance based. Obviously more advanced implementations can make use of munition type, velocity, etc.

The Damage Server can be started in different configurations, optimized for different RTIs. The following configurations are available out of the box:

- **default.config** (any RTI): the default configuration where the DamageServer runs in wall-clock time.
- **pitch-logicaltime.config** (Pitch RTI): the Damage Server runs in logical time (event driven).
- **portico-logicaltime.config** (Portico RTI): the Damage Server runs in logical time (time stepped).

With the configuration for the Pitch RTI the Damage Server sends damage reports and damageState updates at the same logical time as the munition detonation. This means that other federates will see a munition detonation and the resulting damage report and possible damageState update happening at the same logical time.

B.2.2.3 Synopsis

```
docker run <docker options> msaas-nld/damageserver:pi <container options>
```

```
docker run <docker options> msaas-nld/damageserver:po <container options>
```

B.2.2.4 Docker Options

`-p 4000` : socket port number for remote debugging. Optional.

`-e DEBUG=<some value>` : allow a remote debugger to connect via a socket. Optional.

B.2.2.5 Container Options

-h, --help : Optional (no help)

-F, --federation <federation name>: Optional (\$FEDERATIONNAME)

-f, --federate <federate name>: Optional (-)

-L, --logfile <log filename>: Optional (-)

-l, --loglevel <log level>: Optional (SEVERE)

-c, --configfile <configuration filename>: Optional (default.config)

B.2.2.6 Other Information

B.2.2.6.1 Federation Object Models

The Damage Server uses the following FOM modules:

- wall-clock time: RPR_FOM_v2.0_1516-2010.xml, Damage.xml
- logical time: RPR_FOM_v2.0_1516-2010-TSO.xml, Damage.xml

B.2.2.6.1.1 Configuration

An example of the configuration file is shown below. The configured time scheme in this example is WALLCLOCKTIME, so the logical time settings are ignored. At the end of the file are the ranges for damage calculation.

```
# DAMAGE SERVER CONFIGURATION FILE

# -----
# Federation management related configuration
# -----

# directory with foms to use (foms for RO, tsofoms for TSO)
fomDirectory = foms

# number of RTI connect attempts (negative value for indefinite)
connectAttempts = -1

# -----
# Time management related configuration
# -----

# time scheme: LOGICALTIME or WALLCLOCKTIME
time.scheme = WALLCLOCKTIME

# -----
# Logical time settings
# -----

# logical time advancement scheme: NMR, NMRA, TAR, TARA
logicaltime.scheme = NMRA
# lookahead
logicaltime.lookahead = 0
# step size for TAR/TARA scheme
logicaltime.stepsize = 1

# -----
```

```
# Wallclock time settings
# -----

# scaled real time factor
wallclocktime.scale = 1
# start at 0 (true) or at current wallclock time (false)
wallclocktime.isZeroStartTime = true

# -----
# Callback settings
# -----

# RTI Callback scheme used: SINGLE or MULTIPLE
callback.scheme = MULTIPLE
callback.mintime = 1
callback.maxtime = 1

# -----
# Ownership management related configuration
# -----

# attempt to acquire ownership of the damageState attribute
acquireDamageAttribute = false

# -----
# Damage calculation related configuration
# -----

# range (m) for DESTROYED
detonation.destroyedRange = 100
# range (m) for MODERATE_DAMAGE
detonation.moderateDamageRange = 200
# range (m) for SLIGHT_DAMAGE
detonation.slightDamageRange = 300

# -----
# Time related configuration
# -----

# use tag to determine time of spatial as per GRIM 2.0
usetimetag = false
```

B.2.2.7 Example

B.2.2.7.1 Pitch Composition (Wall-Clock Time)

In the following composition the Damage Server is started with the default configuration, i.e., runs wall-clock time. Note that shipsim in this composition is started with all `--fom-modules-dir` allrofoms to use all RO FOM modules instead of the default `RPR_FOM_v2.0_1516-2010-TSO.xml` FOM.

```
version: '2'

services:
  crc:
    image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
    mac_address: 00:18:8B:0D:4F:0B

  web:
    image: app-docker136.hex.tno.nl:443/pitch/web:2.1.1
    ports:
      - "8080:8080"

  shipsim:
    image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
    command: -f ShipSim --fom-modules-dir allrofoms
```

```
damsim:
image: app-docker136.hex.tno.nl:443/msaas-nld/damageserver:pi
command: -f DamageSim
```

B.2.2.7.2 Portico Composition (Logical Time)

In the following composition the federates are started in logical time, using the Portico RTI. With the Portico RTI, one federate application must create the federation execution with all FOM modules before the other federate applications join the federation execution; this is a known issue. The order in which federate applications are started is loosely determined with the LRC_MINSLEEP setting. A better way to manage this is by using a bootstrap federate (for more information see Section B.2.9).

```
version: '2'

services:
  shipsim:
    image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:po
    command: -f ShipSim -timemanaged --fom-modules-dir alltsofoms

  damsim:
    image: app-docker136.hex.tno.nl:443/msaas-nld/damageserver:po
    command: -f DamageSim -c portico-logicaltime.config
    environment:
      - LRC_MINSLEEP=10
```

B.2.2.7.3 Configuration Files

The Damage Server image contains three ready-made configuration files. You can configure the Damage Server with your own configuration file by mounting your configuration file with the Docker volume option:

```
damsim:
image: app-docker136.hex.tno.nl:443/msaas-nld/damageserver:pi
command: -f DamageSim -c myconfig.txt
volumes:
  - ./myconfig.txt:/root/application/myconfig.txt
```

B.2.2.7.4 Additional FOMs

In a similar way additional FOM modules can be added to the default FOM directory of the Damage Server, in this example a Sensor FOM module:

```
damsim:
image: app-docker136.hex.tno.nl:443/msaas-nld/damageserver:pi
command: -f DamageSim
volumes:
  - ./Sensor.xml:/root/application/foms/Sensor.xml
```

B.2.3 EPIC Enhanced Perception and Integrated Control Image (LM)

B.2.3.1 Image

```
app-docker136.hex.tno.nl:443/msaas-usa/epic:dis
```

B.2.3.2 Description

EPIC is a web-based application that provides Common Operating Picture (COP) functionality to exercise controllers and role players of joint staff and coalition training exercises. This image of EPIC provides

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

connectivity to other MSaaS services (ShipSim, VR-Forces, etc.) via a DIS interface. This image relies on the DISWebGateway (provided by open-dis) to forward the DIS traffic to a WebSocket that EPIC is listening on. It also depends on the DIS-HLA Adapter to convert the HLA Objects/Interactions into specific DIS PDUs.

This image of EPIC also has connectivity to the SES Gulf imagery via a WMS imagery layer to provide an enhanced perception of the training area. It is assumed that the SES imagery is running on port 8080. This can be configured using the `-e` option. See the Docker Options section below.

EPIC is released to the MSG-136 group with a limited set of licenses. If you would like to use EPIC for the purposes of this experiment and require a license please contact Robbie Phillips (robbie.phillips@lmco.com). The license agreement can be found [here](#).

B.2.3.3 Synopsis

An example composition to run this image with the ShipSim and vr-forces images. See also: EPIC Compositions at <https://github.com/globalsim/msaas-A/tree/master/composefiles/EPIC>.

```
version: '3'

services:
  dis:
    image: app-docker136.hex.tno.nl:443/pitch/dis:2.6.0L
    command: -auto
    mac_address: 00:18:8B:0D:4F:1B
    environment:
      - DISPLAY=xserver:0
      - MINSLEEP=5

  crc:
    image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.0.0L
    mac_address: 00:18:8B:0D:4F:0B

  web:
    image: app-docker136.hex.tno.nl:443/pitch/web:2.1.1
    ports:
      - "8080:8080"

  shipsim:
    image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
    command: -f ShipSim --fom-modules-dir allrofoms

  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver
    ports:
      - "8081:8080"

  dis-web-gw:
    image: app-docker136.hex.tno.nl:443/msaas-usa/dis-web-gw:epic
    ports:
      - "8282:8282"
      - "8283:8283"

  epic:
    image: app-docker136.hex.tno.nl:443/msaas-usa/epic:dis
    environment:
      - ses_url=http://10.10.10.11:8080/SgjWMS/WMS
      - dis_gw_url=10.10.10.11
    ports:
      - "7070:7070"
```


B.2.3.4 Docker Options

`-e, --env` : Environment Variable options for epic

- `ses_url`: Allows for setting the SES url so EPIC can connect to the WMS service. ** Default is `http://localhost:8080/SgjWMS/WMS` so you only need to specify when using different URL:PORT
- `dis_gw_url`: Specifies the address of the DIS websocket.

Example:

```
-e ses_url="http://192.168.99.1:8081/SgjWMS/WMS"
```

```
-e dis_gw_url="192.168.99.1"
```

B.2.3.5 Container Options

None.

B.2.3.6 Other Information

B.2.3.6.1 Browser Info

EPIC is best used/most tested in Google Chrome using incognito mode. EPIC requires that WebGL is enabled in your browser.

Navigate to `localhost:7070/epic` and login as ExCon using the following credentials:

- User: Exercise Controller.
- Password: admin.

B.2.3.7 Example

N/A.

B.2.4 FEAT Editor Image (TNO)

B.2.4.1 Image

`app-docker136.hex.tno.nl:443/msaas-nld/feat`

B.2.4.2 Description

The MSaaS Registry could not be complete without a dockerized FEAT editor. You can use this image to capture your federation agreements.

For more information see SourceForge FEAT Editor at <https://sourceforge.net/projects/feateditor/>.

B.2.4.3 Synopsis

```
docker run <docker options> msaas-nld/feat:latest <container options>
```

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

B.2.4.4 Docker Options

-e DISPLAY=<X11 display> : X display to use. Required if you want to see anything.

B.2.4.5 Container Options

None.

B.2.4.6 Other Information

N/A.

B.2.4.7 Example

In the following Docker Compose file run.yml the FEAT Editor is started together with an X Server. Start the composition with:

```
docker-compose -f run.yml up
```

Open up a web browser and navigate to

http://DOCKER_HOST:8080/vnc.html

Next login.

Note that the directory where the FEAT Editor starts from is mounted to the Docker Host so that files are saved to persistent storage on the Docker Host.

```
version: '2'

services:
  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver
    ports:
      - "8080:8080"

  feat:
    image: app-docker136.hex.tno.nl:443/msaas-nld/feat
    environment:
      - DISPLAY=xserver:0
    depends_on:
      - xserver
    volumes:
      - ./root
```

B.2.5 Google Chrome Image (TNO)

B.2.5.1 Image

app-docker136.hex.tno.nl:443/library/gc

B.2.5.2 Description

This image holds Google Chrome (GC). The GC GUI can be accessed via an X Server.

This image can be used to display a web UI via an X Server. Although it does not seem logical to display a web UI via an X Server, there are use cases where this approach comes to aid.

One use case is where a web UI connects as a websocket client back to a websocket server. If the web UI also runs in the container network, there is no need to publish the websocket server address outside this network. A firewall between websocket client and server may otherwise block this. Also, data communication between websocket client and server remains local and larger amounts of data (such as map data) can be transferred much quicker than in the case where websocket client and server are connected via a WAN.

B.2.5.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/library/gc <container options>
```

B.2.5.4 Docker Options

`-e DISPLAY=<x display>` : Address of X Server display. **Required.**

B.2.5.5 Container Options

`--homepage <URL>` : set the GC home page

For more options see <https://peter.sh/experiments/chromium-command-line-switches>.

B.2.5.6 Other Information

N/A.

B.2.5.7 Example

Start the composition below named example.yml with the command `docker-compose -f example.yml up`. The GUI of GC can be accessed via the X Server: <http://<HOSTADDRESS>:8080/vnc.html>.

Note that the URL for the GC homepage can also refer to a service in the composition, e.g., homepage <http://epic:7070/epic> for a service named epic.

```
version: '2'

services:
  gc:
    image: app-docker136.hex.tno.nl:443/library/gc
    command: --homepage http://tno.nl
    environment:
      - DISPLAY=xserver:0

  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver
    ports:
      - "8080:8080"
```

B.2.6 Google Earth Image (TNO)

B.2.6.1 Image

`app-docker136.hex.tno.nl:443/library/ge`

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

B.2.6.2 Description

This image holds Google Earth (GE). The GE GUI can be accessed via an X Server.

This image can be used to display KML data from the KML Server, using the address of the KML Server as link name. An example is provided below.

It is possible to start up multiple GE containers, all using the same or different X Servers as display.

B.2.6.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/library/ge <container options>
```

B.2.6.4 Docker Options

`-e DISPLAY=<x display>` : Address of X Server display. **Required.**

B.2.6.5 Container Options

`-h` : display this help and exit.

`-v` : verbose mode.

`-r <refreshtime>` : refresh time in seconds for the GE view (non-negative integer, default 0).

`-f` : fly to the view on start of GE (default: no). The view is defined by the entities supplied via the network link.

`-w <attempts>` : Number of attempts to wait for X Server, before starting GE (non-negative integer, default is indefinite).

`<place name>` : place name (string, default no name).

`<link name>` : network link name (URL, default no URL).

Both the place name and link name must be provided for the `-r` and `-f` options to have effect.

B.2.6.6 Other Information

Google Earth (7.1.1 and as far as known also more recent versions) does not work behind a proxy. It ignores the `HTTP_PROXY` and `HTTPS_PROXY` settings. This is a known issue, see also <https://productforums.google.com/forum/#!topic/maps/CbNfteQnEUI>.

B.2.6.7 Example

Start the composition below named `example1.yml` with the command `docker-compose -f example1.yml up`. The GUI of GE can be accessed via the X Server: <http://<host-address>:8081/vnc.html>.

```
version: '2'

services:
  crc:
```

```
image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.0.0L
mac_address: 00:18:8B:0D:4F:0B

web:
  image: app-docker136.hex.tno.nl:443/pitch/web:2.1.1
  ports:
    - "8080:8080"

kml:
  image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:pi
  command: -f KMLServer

ge:
  image: app-docker136.hex.tno.nl:443/library/ge
  command: -t 5 -f -r 10 Gulf http://kml:8080/kmlserver/entities
  environment:
    - DISPLAY=xserver:0

xserver:
  image: app-docker136.hex.tno.nl:443/library/xserver
  ports:
    - "8081:8080"

shipsim:
  image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
  command: -f ShipSim

shipbackground:
  image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
  command: -f TrafficSim -background background.json -background-count 200
```

A few points to highlight about this example:

- No external port for the `kml` service is declared. The `ge` service accesses the `kml` service via the internal port.
- The URL for the link name (GE) and the icon server (KML Server) use the service name; there is no need to specify hostnames or IP addresses, and it is transparent where in the network the service runs.
- The `ge` service uses a sleeptime to allow the X Server to start first. A better practice is to have the X Server always running.
- Multiple `ge` services may be started, using different options.

B.2.7 KML Server Image (TNO)

B.2.7.1 Image

`app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:pi`

`app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:po`

`app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:none`

The Pitch (pi) and Portico (po) images are based on LRC version 2. The none image (none) can be mounted as a volume to either one of the LRC base images.

B.2.7.2 Description

This KML Server generates KML data for display in Google Earth or other KML enabled clients. Both RPR-FOM entities and Sensor FOM tracks can be displayed in two different layers in Google Earth. The KML Server can also be used as an Icon Server, serving icons for entity types.

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

The KML Server can be started in different configurations, optimized for different RTIs. The following configurations are available out of the box:

- **default.config** (any RTI): the default configuration where the KML Server runs in wallclock time (RO RPR-FOM).
- **pitch-logicaltime.config** (Pitch RTI): the KML Server runs in logical time (event driven, TSO RPR-FOM).
- **portico-logicaltime.config** (Portico RTI): the KML Server runs in logical time (time stepped, TSO RPR-FOM).

B.2.7.3 Synopsis

```
docker run <docker options> msaas-nld/kmlserver:pi <container options>
```

```
docker run <docker options> msaas-nld/kmlserver:po <container options>
```

B.2.7.4 Docker Options

-p <port number>:<kml port number>: Google Earth port number. **Required.**

-p <port number>:4000: socket port number for remote debugging. Optional.

-e DEBUG=<some value>: allow a remote debugger to connect via a socket. Optional.

B.2.7.5 Container Options

-h, --help: Optional (no help)

-F, --federation <federation name>: Optional (\$FEDERATIONNAME)

-f, --federate <federate name>: Optional (-)

-L, --logfile <log filename>: Optional (-)

-l, --loglevel <log level>: Optional (SEVERE)

-c, --configfile <configuration filename>: Optional (default.config)

-i, --iconurl <icon server> : URL of the Icon Server. Optional (http://<hostname>:8080/kmlserver), where hostname is the hostname of the icon server container

-k, --kmlport <kml port number>: KML Server port number. Optional (8080)

B.2.7.6 Other Information

B.2.7.6.1 Federation Object Models

The KML Server uses the following FOM modules:

- wall-clock time: RPR_FOM_v2.0_1516-2010.xml, Sensor.xml
- logical time: RPR_FOM_v2.0_1516-2010-TSO.xml, Sensor.xml

B.2.7.6.2 Icon Server

Each icon in the KML file generated by the KML Server is identified by a URL. The syntax of the URL for an entity and track are as follows:

```
<icon server>/<force id>/<entity type>  
<icon server>/tracks/<track identification>/<track classification>
```

And where:

```
<icon server> ::= http://<host>:<port> [/<base>]
```

KML Server has an icon server built in and the base of this icon server is by definition kmlserver where:

- `<force id>` : 0..3 (Other, Friendly, Opposing, Neutral).
- `<entity type>` : a sequence of 7 digits separated by dots (Entitykind, Domain, CountryCode, Category, Subcategory, Specific, Extra).
- `<track identification>` : 0..6 (pending, unknown, assumed friend, friend, neutral, suspect, hostile).
- `<track classification>` : see entity type.

B.2.7.6.3 Google Earth Configuration

To view entities in Google Earth, add a Google Earth network place with one of the two following URLs:

```
http://<host>:<port number>/kmlserver/entities
```

To view sensor tracks in Google Earth, add a Google Earth network place with the following URL:

```
http://<host>:<port number>/kmlserver/tracks
```

B.2.7.6.4 Proxy

All URLs for the KML Server have as base the name kmlserver in the URL. A proxy can redirect all KML Server requests using this base name.

B.2.7.7 Example

B.2.7.7.1 Pitch Composition (Wall-Clock Time)

Example of a Pitch RTI based federation. The UI of the CRC and GE can be viewed via a webbrowser: <http://<HOSTADDRESS>:8080/vnc.html>.

```
version: '2'  
  
services:  
  crc:  
    image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L  
    mac_address: 00:18:8B:0D:4F:0B  
    environment:  
      - DISPLAY=xserver:0  
  
  kml:  
    image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:pi  
    command: -f KMLServer
```

```
shipsim:
  image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
  command: -f ShipSim --fom-modules-dir allrofoms

ge:
  image: app-docker136.hex.tno.nl:443/library/ge
  command: -f -r 10 Gulf http://kml:8080/kmlserver/entities
  environment:
    - DISPLAY=xserver:0

xserver:
  image: app-docker136.hex.tno.nl:443/library/xserver
  ports:
    - "8080:8080"
```

B.2.7.7.2 Portico Composition (Wall-Clock Time)

Example of a Portico RTI based federation, using a bootstrap federate called master to manage the creation of the federation execution with all FOM modules. The UI of GE can be viewed via a webbrowser: <http://<HOSTADDRESS>:8080/vnc.html>.

```
version: '2'

services:
  master:
    image: app-docker136.hex.tno.nl:443/msaas-nld/start:po
    command: -p 6666 -d allrofoms

  kml:
    image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:po
    command: -f KMLServer
    environment:
      - LRC_MASTERADDRESS=master:6666

  shipsim:
    image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:po
    command: -f ShipSim --fom-modules-dir foms
    environment:
      - LRC_MASTERADDRESS=master:6666

  ge:
    image: app-docker136.hex.tno.nl:443/library/ge
    command: -f -r 10 Gulf http://kml:8080/kmlserver/entities
    environment:
      - DISPLAY=xserver:0

  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver
    ports:
      - "8080:8080"
```

B.2.7.7.3 Icon Server

Example of a Pitch RTI based federation, using a separate icon server. The UI of the CRC and GE can be viewed via a webbrowser: <http://<HOSTADDRESS>:8080/vnc.html>.

```
version: '2'

services:
  crc:
    image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
    mac_address: 00:18:8B:0D:4F:0B
    environment:
```



```
- DISPLAY=xserver:0

kml:
  image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:pi
  command: -f KMLServer -iconurl http://icon:80/app6b-symbolpng

icon:
  image: app-docker136.hex.tno.nl:443/msaas-dnk/disenumerationsymbolservice:2.2

shipsim:
  image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
  command: -f ShipSim --fom-modules-dir allrofoms

ge:
  image: app-docker136.hex.tno.nl:443/library/ge
  command: -f -r 10 Gulf http://kml:8080/kmlserver/entities
  environment:
    - DISPLAY=xserver:0

xserver:
  image: app-docker136.hex.tno.nl:443/library/xserver
  ports:
    - "8080:8080"
```

B.2.7.7.4 Remote Google Earth Client

Example of a Pitch RTI based federation, using a remote Google Earth client. Entities can be displayed in the remote Google Earth client using the following network place for entities:

```
http://10.10.10.11:8081/kmlserver/entities
```

The address 10.10.10.11 should be replaced by your Docker Host address.

```
version: '2'

services:
  crc:
    image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
    mac_address: 00:18:8B:0D:4F:0B
    environment:
      - DISPLAY=xserver:0

  kml:
    image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:pi
    command: -f KMLServer -iconurl http://10.10.10.11:8081/kmlserver
    ports:
      - "8081:8080"

  shipsim:
    image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
    command: -f ShipSim --fom-modules-dir allrofoms

  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver
    ports:
      - "8080:8080"
```

B.2.8 Logger Service Image (IFAD)

B.2.8.1 Image

```
app-docker136.hex.tno.nl:443/msaas-dnk/ifadloggerservice:1.1-alpine
```

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

B.2.8.2 Description

The IFAD Logger Service image provides a Web Client, an HTTP interface and a download portal for the *IFAD Logger Server v4.0*. The *IFAD Logger Server* is a logging server supporting record/playback of DIS messages over a network.

The Web Client provides a standard set of control functions for the *IFAD Logger Server* while the HTTP service interface gives an increased level of control.

Note: The *IFAD Logger Server* is controlled via the XML-RPC protocol whereas the IFAD Logger Service Image uses an HTTP approach. Documentation for the XML-RPC interface is not provided in this context.

B.2.8.3 Synopsis

```
Docker run <docker options> app-docker136.hex.tno.nl/msaas-dnk/  
ifadloggerservice:1.1-alpine
```

B.2.8.4 Docker Options

Recommended:

-d: Run container in background

Required:

-p <port1>:80

-p <port2>:8080

Optional:

-p <port3>:50000

B.2.8.5 Container Options

Nil.

B.2.8.6 Service Interface

- /<port1>
- /<port1>/storage
- /<port1>/about (same as /<port1>/help)
- /<port1>/<instruction>
- /<port1>/<instruction>[/<value1>/<value2>/...]
- /<port2>/lwc (same as /<port2>/lwc/logger)
- /<port2>/lwca (same as /<port2>/lwca/logger)

Where,

- **about** and **help** points to the manual for the HTTP interface.
- **storage** points to the download portal.
- **port1** is publishing container port 80 for the HTTP interface.

- **port2** is publishing container port 8080 for the Web Clients.
- **port3** is publishing container port 50000 for the IFAD Logger XML-RPC server.
- **instruction** is one of deletelog, getconnectionparameters, getcurrentlogid, getelapsedplaybacktime, killloggerservice, listlogs, load, loadwithallinfo, pause, playback, productversion, record, resume, save, setupdisparameters, setupdisparameters, startloggerservice, status, stop, storage, viewfulllogininfo and viewsimplelogininfo.
- **value1, value2, ...** are required values for a subset of the above instructions (see `<port1>/about` for further details)
- **lwc** points to a Web Client for the containerized IFAD Logger server
- **lwca** points to an autonomous Web Client for an IFAD Logger server (v4.0)

B.2.8.7 Examples

Start the services on e.g., port 9080 and 90 on localhost.

```
docker run -d -p 9080:8080 -p 90:80 -d app-docker136.hex.tno.nl/msaas-dnk/ifadloggerservice:1.1-alpine
```

The integrated Web Client is at

`http://localhost:9080/lwc`

A service manual for the HTTP interface is found at

`http://localhost:90/about`

where parameters such as logID and XML-RPC_port are described.

To request the status of the IFAD Logger Server, use

`http://localhost:90/status`

The logger service can be (re)started with new parameters – for example XML-RPC_port=50000, Broadcast_address=10.11.12.13, Exercise_ID=47, Object_time_out=3, Site_ID=56, Application_ID=545 and Heatbeat=0 – via the request:

`localhost:90/startloggerservice/50000/10.11.12.13/47/3/56/545/0`

Note: It is advised to always use XML-RPC_port=50000.

To change connection parameters – say Port=23500, Broadcast_address=255.255.255.255, Exercise_ID=1 and Object_time_out=10 – use,

`localhost:90/setupdisparameters/23500/255.255.255.255/1/10`

Connection parameters are listed via,

`localhost:90/getconnectionparameters`

To start a logger recording, use

`localhost:90/record`

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

A recording is stopped with,

```
localhost:90/stop
```

A recording can be saved with the request

```
localhost:90/save
```

The /save request changes the current logID as stressed by

```
localhost:90/getcurrentlogid
```

Recordings saved under the DIS catalog are found via

```
localhost:90/listlogs/DIS
```

Recordings are downloaded via the portal

```
localhost:90/storage/
```

The logger service plays back is from currently loaded logID. To play back from beginning, use

```
localhost:90/playback/0
```

It is possible to pause and resume via

```
localhost:90/pause
```

respectively

```
localhost:90/resume
```

To play back from different recording, first stop any playback or recording

```
localhost:90/stop
```

then load a new logID via the /load request,

```
localhost:90/load/<name>/DIS
```

where name is in the response of the /listlogs/DIS request.

B.2.8.8 Other Information

None.

B.2.9 LRC Base Image (TNO)

B.2.9.1 Image

LRC version 1:

```
app-docker136.hex.tno.nl:443/pitch/lrc:5.3.0.0-<platform>
```

```
app-docker136.hex.tno.nl:443/portico/lrc:nightly-20160528-<platform>
```

```
app-docker136.hex.tno.nl:443/vtmak/ma-lrc-base:latest
```

- platform for Pitch: alpine, debian, centos (version 6)
- platform for Portico: alpine, debian
- platform for MaK: centos

LRC version 2:

app-docker136.hex.tno.nl:443/pitch/lrc:2-5.3.2.1-<platform>

app-docker136.hex.tno.nl:443/portico/lrc:2-nightly-20160528-<platform>

- platform for Pitch: alpine, debian, centos6, centos7
- platform for Portico: alpine, debian

B.2.9.2 Description

These images are LRC base images for HLA federate applications. An HLA federate application is the federate code without any RTI libraries. The LRC base images are based on Alpine Linux, Debian or CentOS, and include a Java JRE. Currently supported RTIs: Pitch, Portico, MaK.

Two approaches – design patterns – for the containerization of HLA federate applications using an LRC base image are described in SISO SIW paper 2016-SIW-031, available from the SISO Digital Library. The patterns are named:

- Containerization via extension; and
- Containerization via composition.

Please read the SIW paper to learn more.

B.2.9.2.1 LRC Settings

LRC settings are environment variables used to configure the LRC (see Table B-1). There are general environment variables applicable to any RTI, and RTI specific variables.

B.2.9.2.1.1 General

Table B-1: Environment Variables.

LRC Version	Environment Variable	Description	Default if Not Specified
1	ENTRYPOINT	Shell script to start the federate application	Container exit
2+1	LRC_MASTERHOST	Master hostname or host address	i)
1	LRC_MASTERPORT	Master port number	ii)
1	MINSLEEP	Minimum sleep time in integer seconds before the LRC base image starts the federate application	0
1	MAXSLEEP	Maximum sleep time in integer seconds before the LRC base image starts the federate application	\$MINSLEEP

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

LRC Version	Environment Variable	Description	Default if Not Specified
2	LRC_ENTRYPOINT	Shell script to start the federate application	Container exit
2	LRC_MASTERADDRESS	Master hostname and portnumber as <name>:<port>	iii)
2	LRC_MASTERATTEMPTS	Number of attempts to connect to master	-1
2	LRC_MINSLEEP	Minimum sleep time in integer seconds before the LRC base image starts the federate application	0
2	LRC_MAXSLEEP	Maximum sleep time in integer seconds before the LRC base image starts the federate application	\$LRC_MINSLEEP

i) Dependent on LRC base image:

- Pitch: if LRC_MASTERHOST is unset or empty string then defaults to \$PITCH_CRCHOST; and
- Portico: unset for Portico.

ii) Dependent on LRC base image:

- Pitch: if LRC_MASTERPORT is unset or empty string then defaults to \$PITCH_CRCPORT; and
- Portico: unset for Portico.

iii) Dependent on LRC base image:

- Pitch: if LRC_MASTERADDRESS is unset or empty string then defaults to \$PITCH_CRCADDRESS; and
- Portico: unset for Portico.

If LRC_MASTERHOST and LRC_MASTERPORT (for version 1) or LRC_MASTERADDRESS (for Version 2) is set to a non-empty string, then the LRC base image does a maximum of LRC_MASTERATTEMPTS attempts to connect to the master address, before starting the federate application. The design pattern is that a master component bootstraps on the master host, creates/joins the federation execution, and opens the master port when ready. Once the port is open, an LRC base image can then start the federate application. For a Pitch LRC base image the Pitch CRC is by default the master component; this can be changed by providing other values for master host/port.

For version 1 LRC_MASTERATTEMPTS can be regarded as -1. For version 2 LRC_MASTERATTEMPTS is the number of attempts that is made to connect to the master address. If this is a negative value, then the number of attempts is indefinite. If this is a zero value, then no attempt is made. If this is a positive value, then this number of attempts is made. After the specified number of attempts is made the Federate Application specified in LRC_ENTRYPOINT is started.

The MIN-MAX sleep period applies from the point in time where the master host/port is open (if master host/port set), or from the point in time where the container is started (if master host/port unset).

B.2.9.2.1.2 Pitch

Table B-2 outlines the environment variables for Pitch.

Table B-2: Environment Variables.

LRC Version	Environment Variable	Description	Default if Not Specified
1	PITCH_CRCHOST	CRC hostname or hostaddress.	crc
1	PITCH_CRCPORT	CRC portnumber.	8989
2	PITCH_CRCADDRESS	CRC address in the format of <host>:<port> or <crc-nickname>@<boost host>:<boost port>	crc:8989
2	PITCH_BOOSTADAPTER	The network adapter that the LRC should use for the Booster network.	Current setting (all)
2	PITCH_ADVERTISE_ADDRESS	Use this address to advertise the LRC. The format is <host address>[:<mintcpport>]-<maxtcpport>[:<minudpport>]-<maxudpport>]]	Host address
	PITCH_LRCADAPTER	The network adapter that the LRC should use.	Current setting (all)
	PITCH_ENABLETRACE	Set to any value to enable. Enable RTI and Federate Ambassador tracing to console.	No tracing

When using PITCH_ADVERTISE_ADDRESS with a port range, make sure that the same port range is also provided in the container -p option. For example:

```
docker run \
  -e PITCH_ADVERTISE_ADDRESS=10.10.10.11:6100-6110 \
  -p 6100-6110:6100-6110 \
  app-docker136.hex.tno.nl:443/imagename
```

B.2.9.2.1.3 Portico

Table B-3 outlines environment variables for Portico.

Table B-3: Environment Variables.

LRC Version	Environment Variable	Description	Default if Not Specified
	PORTICO_RTI_RID_FILE	File path to the Portico RID file	\$LRC_HOME/RTI.rid
	PORTICO_LOGLEVEL	Specify the level that Portico will log at. Valid values are: TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF.	Current setting (WARN)
	PORTICO_UNIQUEFEDERATIONAMES	Ensure that all federates in a federation have unique names. When false, Portico will change the requested name from “name” to “name (handle)” thus making it Unique. Valid values are: true, false.	Current setting (true)
	PORTICO_LRCADAPTER	The network adapter (network interface) that the LRC should use. The name must be an exact match.	Whatever JGroups selects

B.2.9.2.1.4 MÄK RTI

Table B-4 outlines environment variables for Portico.

Table B-4: Environment Variables.

LRC Version	Environment Variable	Description	Default if Not Specified
	MAK_RTI_RID_FILE	The path to a custom RID file. Note that such a custom file needs to conform to certain RID properties used by the <code>rtiexec</code> . Two RID files are supplied in the <code>mak-rti-baseimage</code> at <code>/etc/makrti/rid.mtl</code> and <code>/etc/makrti/rid-lm.mtl</code> . See MÄK <code>rtiexec</code> Image at https://github.com/globalsim/msaas-A/wiki/M%C3%84K-rtiexec-Image for more details.	<code>/etc/makrti/rid.mtl</code> as set by the <code>mak-rti-base image</code> .
	MAK_RTIEXEC_ADDR	The IP address of the endpoint supplying the <code>rtiexec</code> (or, more accurately, the IP address of an RTI forwarder, which, by default, runs with the <code>rtiexec</code>).	Not set. See environment variable <code>MAK_RTIEXEC_HOSTNAME</code> .

LRC Version	Environment Variable	Description	Default if Not Specified
	MAK_RTIEEXEC_HOSTNAME	The hostname (container name) of the endpoint running the <code>rtiexec</code> (see <code>MAK_RTIEEXEC_ADDR</code>). If <code>MAK_RTIEEXEC_ADDR</code> is not set, the IP address of the endpoint running the <code>rtiexec</code> will be looked up within <code>/etc/hosts</code> using the value of the <code>MAK_RTIEEXEC_HOSTNAME</code> environment variable. It is necessary to start a container using <code>ma-lrc-base</code> with a <code>link</code> option so that an entry is written into <code>/etc/hosts</code> . The IP address is written into the RID file (located via environment variable <code>RTI_RID_FILE</code>) for parameter <code>RTI_tcpForwarderAddr</code> .	<code>rtiexec</code>
	MAK_LOG_LEVEL	Change the level of logging detail generated by the LRC. Values in the range 0 – 4 are valid with 0 being no logging and 4 being the most detailed	2
	MAK_LOGFILE_DIR	Specify a directory into which LRC log files will be written. This is most useful if it is a volume mount so that the log file becomes visible on the host system	Working directory of the federate
	MAK_FEDERATE_LOGFILE	The name of the log file to be written by the LRC. The file is written into <code>MAK_LOGFILE_DIR</code> . Logging to file is not enabled unless this environment variable is set.	Not set

B.2.9.2.1.5 Java Federate

For a Java federate `LRC_CLASSPATH` must be added to the Java `CLASSPATH` environment variable or to the `java -cp` command line option. In addition, for the `MAK` LRC, the contents of `LRC_LIBRARYPATH` must be added to the `LD_LIBRARY_PATH` environment variable. This is required as the `MAK` RTI implementation of the Java HLA Interface Specification is a binding layer over a C++ implementation and those C++ libraries need to be on the `LD_LIBRARY_PATH` at runtime.

B.2.9.2.1.6 C++ Federate

For a C++ federate `LRC_LIBRARYPATH` must be added to the `LD_LIBRARY_PATH` environment variable.

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

B.2.9.2.2 Federate Application Settings

Federate settings are environment variables to use by or to initialize the federate application (see Table B-5).

Table B-5: Environment Variables.

LRC Version	Environment Variable	Description
	FEDERATIONNAME	The federate shall use this name as the name of the federation to join. This variable is by default set to “TheWorld”. The setting can be overridden by the user by setting the variable to some value.
1	LRC_CONNECTATTEMPTS	The federate shall use this value as the number of attempts to connect to the RTI, with one attempt per second. Default is 0, meaning attempt indefinite.
	LRC_HOME	Home directory in which the LRC include files and libraries are stored.
	LRC_CLASSPATH	RTI class path files for Java applications.
	LRC_LIBRARYPATH	RTI library search path for shared objects for C++ applications.
	JAVA_HOME	Installation directory of the Java Runtime Environment.
	LRC_VERSION	Version identification of the LRC image.

B.2.9.3 Other Information

B.2.9.3.1 Containerization via Extension

Linking by extension takes the traditional approach of building a federate application image based on an LRC image. In this pattern, the container image of the federate application is built using the Dockerfile FROM instruction. The FROM instruction sets the base image for the federate application, in this case an LRC image. The LRC is combined with the federate application at image build time. As an example, consider building a federate application image based on msaas/pi-lrc-base:5.3.0.0-alpine, a small image based on Alpine Linux, with Pitch LRC version 5.3.0.0 and Java Runtime Engine 8 installed. The Dockerfile for building the federate application image looks as follows:

```
FROM app-docker136.hex.tno.nl:443/pitch/lrc:5.3.0.0-alpine

#Install application
COPY ./application /root/application

#Set entrypoint
ENV ENTRYPOINT=/root/application/start.sh
```

This Dockerfile does the following three things:

- It sets the base image to `pitch/lrc:5.3.0.0-alpine`;
- It copies the application directory, containing the federate application, to the filesystem of the container at `/root/application`; and
- It defines the environment variable `ENTRYPOINT`, setting it to the name of a shell script that is used to start the federate application (in this example `start.sh`).

The installation directory can be anywhere in the container's filesystem, so long as the `ENTRYPOINT` environment variable points to a shell script that starts the federate application.

The start script is federate application specific and will receive all the container command line options (if any) from the callee (the `launch.sh` script in the base LRC image). For a Java federate application, the start script may be as follows:

```
#!/bin/sh
java -cp MyApplication.jar:$LRC_CLASSPATH myapplication.Main $@
```

The environment variable `LRC_CLASSPATH` is set by the base LRC image and has all the necessary LRC jar files for interacting with the RTI (equivalent environment variables would be set in the LRC base image for sourcing C++ shared object libraries). The command line options are passed on to the federate application via `"$@"`.

B.2.9.3.2 Gracefully Stopping a Containerized HLA Federate Application

When a container is stopped (with the `docker stop` command) the signal `SIGTERM` is sent to the process with PID 1 running inside the container. If this process does not terminate within 10 seconds, Docker will kill the process with the signal `SIGKILL`. This means that an HLA federate application will “disappear” abruptly from the federation execution and any resources that the container process holds are not released in an orderly fashion. Thus, to gracefully stop a container and its containing process(es), the process with PID 1 must handle the `SIGTERM` signal.

As mentioned earlier the federate application is started through the shell script defined by the environment variable `ENTRYPOINT`. The LRC `launch.sh` script ensures that this shell script runs in a shell with PID 1.

When the start script listed in the previous section is used to start the application, the application will not be terminated by the `SIGTERM`. This is because the application is actually executed in a new shell, forked from the shell with PID 1. The new shell will not receive the `SIGTERM` signal.

For a Java federate application, the following start script can be used to forward the `SIGTERM` signal to the application, running inside a Java Virtual Machine (JVM):

```
#!/bin/sh

# Initialise the PID of the application
pid=0

# define the SIGTERM-handler
term_handler() {
    echo 'Handler called'
    if [ $pid -ne 0 ]; then
        kill -SIGTERM "$pid"
        wait "$pid"
    fi
    exit 143; # 128 + 15 -- SIGTERM
}
```

```
}

# on signal execute the specified handler
trap 'term_handler' SIGTERM

# run application in the background and set the PID
java -cp MyApplication.jar:$LRC_CLASSPATH myapplication.Main $@ &
pid="$!"

wait "$pid"
```

The start script forwards the SIGTERM to the JVM, where it is up to the application running inside the JVM to handle the SIGTERM. The application running inside the JVM should use a Shutdown Hook to catch the shutdown of the JVM. For more information on handling SIGTERM and JVM shutdown, see Shutdown Hook.

B.2.9.4 Example

None.

B.2.10 MSaaS Portal Image (TNO)

B.2.10.1 Image

app-docker136.hex.tno.nl:443/msaas-nld/portal

B.2.10.2 Description

This image provides an implementation of an MSaaS portal. The portal provides access to services that are organized in user configurable tabs in the portal. Most common services are Docker Compose UI and Portainer; an example of a portal configuration is provided below.

B.2.10.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/msaas-nld/portal
<container options>
```

B.2.10.4 Docker Options

-p <port>:8080 : port number where portal can be accessed on.

-e PORTAL_TITLE=<name> : title of the web page (optional; default is MSaaS Portal)

-e PORTAL_TAB<n>_NAME=<tab name> : name of the tab

-e PORTAL_TAB<n>_ADDR=<url> : address of the service under this tab

Where <n> is the tab index. Tabs are indexed from 0 onwards.

B.2.10.5 Container Options

None.

B.2.10.6 Other Information

N/A.

B.2.10.7 Example

Start a MSaaS portal that includes Docker Compose UI, Weave Scope, and Portainer using the docker-compose file shown below. The host address `PORTAL_ADDR` of the portal is passed in to the composition as an environment variable.

The source files for this example can be found in the code repository, at this location.

The composition can be started with the following command:

```
PORTAL_ADDR=`hostname -i` docker-compose up
```

Next, in your web browser, navigate to `http://$PORTAL_ADDR:4000` and Figure B-1 should appear:

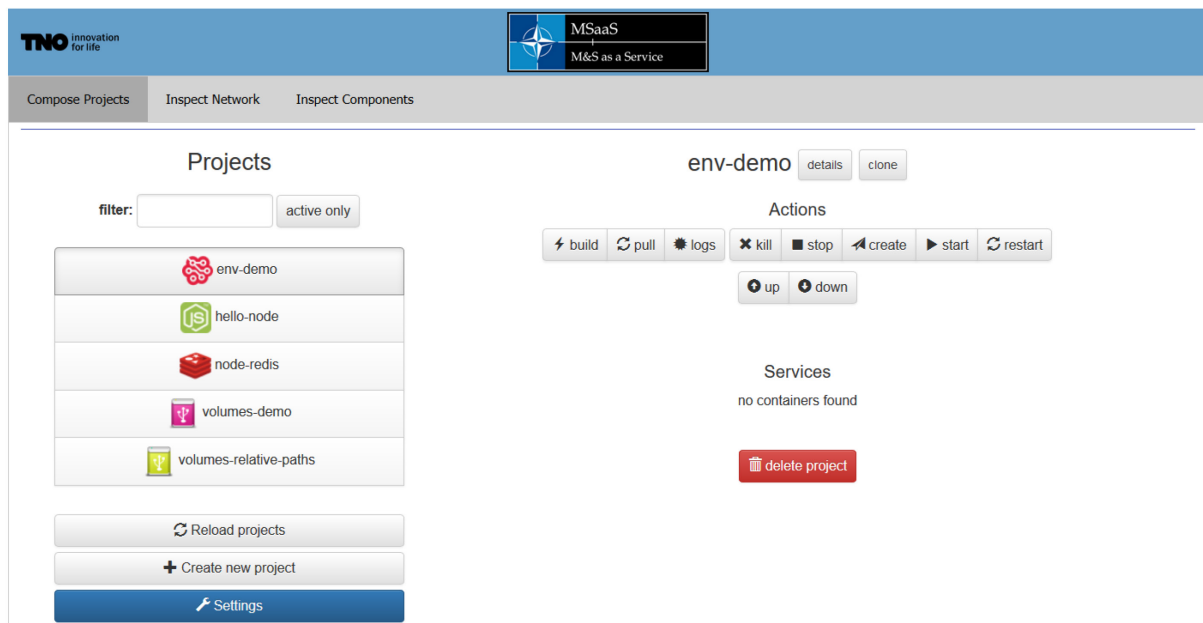


Figure B-1: MSaaS Portal UI.

The composition is as follows:

```
version: '2'

services:
  portal:
    image: app-docker136.hex.tno.nl:443/msaas-nld/portal
    ports:
      - "4000:8080"
    environment:
      - PORTAL_TITLE=My Portal
      - PORTAL_TAB0_NAME=Compose Projects
      - PORTAL_TAB0_ADDR=http://$PORTAL_ADDR:4001
      - PORTAL_TAB1_NAME=Inspect Network
      - PORTAL_TAB1_ADDR=http://$PORTAL_ADDR:4040
      - PORTAL_TAB2_NAME=Inspect Components
      - PORTAL_TAB2_ADDR=http://$PORTAL_ADDR:4002
```

```
composeui:
  image: app-docker136.hex.tno.nl:443/msaas-nld/docker-compose-ui:1.8.1
  ports:
    - "4001:5000"
  volumes:
    - ./projects:/opt/docker-compose-projects
    - $HOME/.docker/config.json:/root/.docker/config.json:ro
    - /var/run/docker.sock:/var/run/docker.sock

scope:
  image: weaveworks/scope
  command: --probe.docker=true
  privileged: true
  container_name: weavescope
  network_mode: "host"
  pid: "host"
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock

portainer:
  image: portainer/portainer
  command: -H unix:///var/run/docker.sock --no-auth
  ports:
    - "4002:9000"
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
```

B.2.11 Munition Server Image (AUS, DST Group)

B.2.11.1 Image

app-docker136.hex.tno.nl:443/msaas-aus/munition:none

app-docker136.hex.tno.nl:443/msaas-aus/munition:pi

app-docker136.hex.tno.nl:443/msaas-aus/munition:po

The Pitch (pi) and Portico (po) images are based on LRC version 2. The none image (none) can be mounted as a volume to either one of the LRC base images.

B.2.11.2 Description

Contains a federate that receives WeaponFire interactions, models the flight of a fired Munition, and sends a Detonation interaction once the Munition reaches the target.

At present, the munition modelling is solely event based. At WeaponFire, the distance from source to target is calculated. This distance is used, along with a default munition speed of 200 m/s, to determine the time the munition would take to get from source to target. A Detonation interaction is sent after that amount of time has elapsed.

B.2.11.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/msaas-aus/munition:pi
<container options>
```

B.2.11.4 Docker Options

None.

B.2.11.5 Container Options

```
-F, --federation-name  
-f, --federate-name  
--fom-modules  
--fom-modules-dir  
-r, --realtime-rate  
--timemanaged  
--timestep  
--lookahead
```

B.2.11.6 Other Information

None.

B.2.11.7 Example

None.

B.2.12 MÄK License Manager Image (AUS, DST Group)**B.2.12.1 Image**

app-docker136.hex.tno.nl:443/vtmak/ma-lm

B.2.12.2 Description

Provides the MÄK License Manager server that serves licenses to MÄK Technologies products. Most notably, these licenses are required when federates connect to the rtiexec.

B.2.12.3 Synopsis

```
docker run <docker options> --mac-address 02:C9:96:5B:12:B7 msaas-  
registry.cloudapp.net:5000/msaas/ma-lm
```

B.2.12.4 Docker Options

```
--mac-address 02:C9:96:5B:12:B7
```

The license included with this License Manager image is locked to MAC address 02:C9:96:5B:12:B7. Therefore, the container needs to be started with this MAC address.

```
--name <container name>
```

It is recommended that a container name be specified since it is needed when starting other MÄK Technologies products. For example, the MÄK rtiexec Image requires a link mapping from an internal license server name (maklm by default) to the actual name of the container running the License Manager.

B.2.12.5 Container Options

Nil.

B.2.12.6 Other Information

Nil.

B.2.12.7 Example

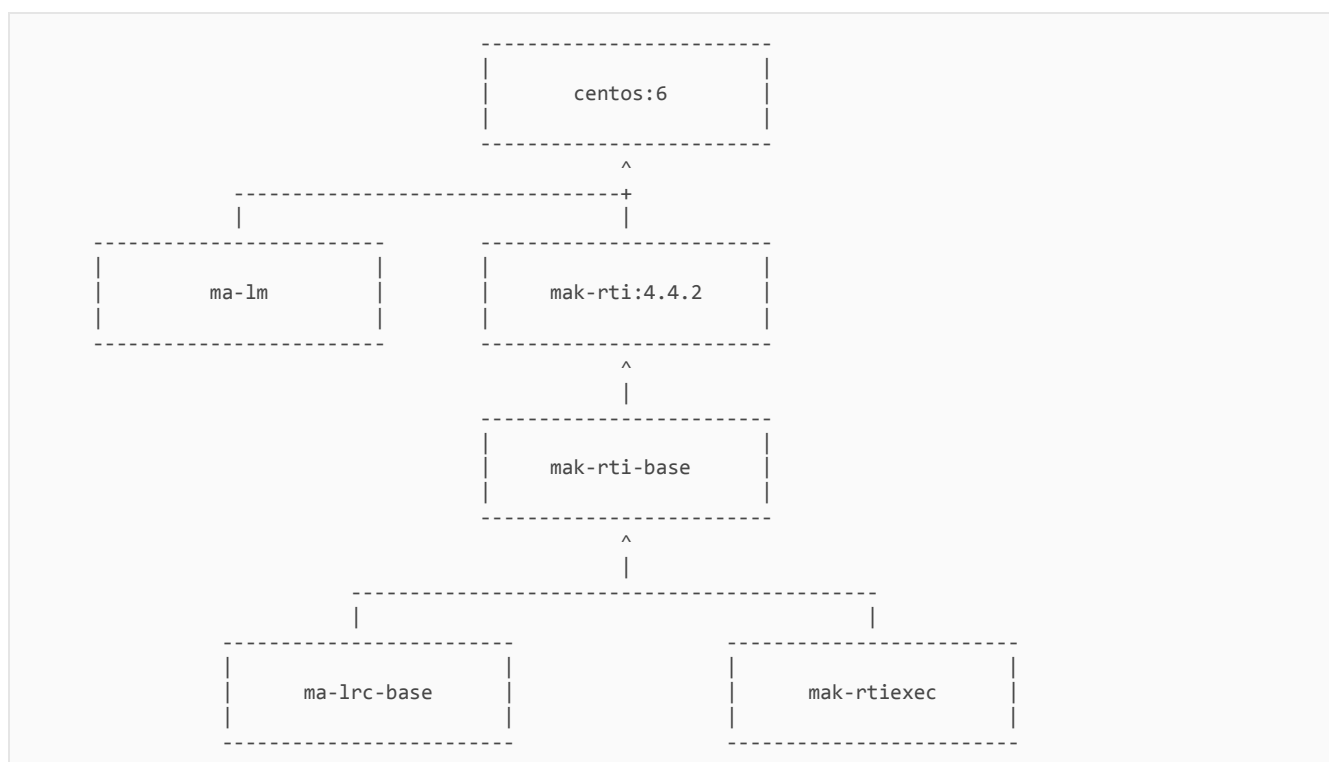
```
docker run -d --name maklm --mac-address 02:C9:96:5B:12:B7 app-  
docker136.hex.tno.nl:443/vtmak/ma-lm
```

B.2.13 MÄK RTI Image Structure (AUS, DST Group)

B.2.13.1 Overview

The MÄK RTI consists of a number of components: the rtiexec, RTI forwarder, RTI assistant and LRC. In addition, licensing requires a license manager server. The image structure composing these elements is illustrated below. The image names are all in the namespace app-docker136.hex.tno.nl:443/msaas/.

The image tree is based from CentOS 6.



B.2.13.2 mak-rti:4.4.2

Image `mak-rti:4.4.2` adds MÄK RTI 4.4.2 and installs dependencies necessary for running the rtiexec. No entrypoint is defined so this can be used to create a terminal container. Environment variable `MAK_RTIDIR` is set to the installation directory.

B.2.13.3 ma-lm

Image `ma-lm` provides the MÄK License Manager. The installed license is locked to MAC address `02:C9:96:5B:12:B7`.

B.2.13.4 mak-rti-base

Image `mak-rti-base` adds two RID files for MSG-136 federations.

- `/etc/makrti/rid.mtl`: enables full HLA compliance, forces connection configuration via the RID file, and enables the `rtiexec` to perform license management for connected federates (this removes the requirement for federates to connect to the license manager).
- `/etc/makrti/rid-lm.mtl`: is the same as `/etc/makrti/rid.mtl` except that licence management for federates is not supported by the `rtiexec`. Instead, federates need to point to the MÄK Licence Manager by assigning a value to the environment variable `MAKLMGRD_LICENSE_FILE`. The value is of the form `@hostname` where `hostname` is the name of the computer running the license manager.

The `mak-rti-base` image sets environment variable `RTI_RID_FILE` to point to the `/etc/makrti/rid.mtl` RID file and defines `RTI_ASSISTANT_DISABLE` to turn off the MÄK RTI Assistant.

B.2.13.5 `mak-rti-base`

Image `mak-rti-base` adds two RID files for MSG-136 federations.

- `/etc/makrti/rid.mtl`: enables full HLA compliance, forces connection configuration via the RID file, and enables the `rtiexec` to perform license management for connected federates (this removes the requirement for federates to connect to the license manager).
- `/etc/makrti/rid-lm.mtl`: is the same as `/etc/makrti/rid.mtl` except that licence management for federates is not supported by the `rtiexec`. Instead, federates need to point to the MÄK Licence Manager by assigning a value to the environment variable `MAKLMGRD_LICENSE_FILE`. The value is of the form `@hostname` where `hostname` is the name of the computer running the license manager.

The `mak-rti-base` image sets environment variable `RTI_RID_FILE` to point to the `/etc/makrti/rid.mtl` RID file and defines `RTI_ASSISTANT_DISABLE` to turn off the MÄK RTI Assistant.

B.2.13.6 `mak-rtiexec`

Image `mak-rtiexec` adds an entrypoint to launch the `rtiexec`. Launching is performed with shell script (`/root/rtiexec/rtexec.sh`) via `supervisord` (since the `rtiexec` exits immediately and causes a started container to stop immediately). Environment variable `MAKLMGRD_LICENSE_FILE` is set to `@maklm` requiring the `rtiexec` container to be started with a link definition mapping `maklm` to the name of the container running the license manager (the name of the licence manager can be overridden by specifying a new value for `MAKLMGRD_LICENSE_FILE` but a `link` statement is still required so that the name of the license manager is inserted into the `/etc/hosts` file of the `rtiexec` container).

B.2.13.7 `ma-lrc-base`

Image `ma-lrc-base` provides the base image for launching federates. It includes the standard `launch.sh` and a MÄK RTI specific `settings.sh`. Most notably, the `settings.sh` will process environment variables to identify the IP address of the container running the `rtiexec` (more accurately, it is the IP address of the container running the RTI Forwarder, which can be started by the `rtiexec`). This can be done by explicitly setting the IP address via environment variable `MAK_RTIEXEC_ADDR`, but, this is not robust since container IP addresses are dynamically assigned. The alternative is to set a `hostname` for the `rtiexec` container via environment variable `MAK_RTIEXEC_HOSTNAME` (the default name is `rtiexec`). However, the MÄK RTI seems to not like using `hostnames` for identifying endpoints so `settings.sh` looks for the IP address of `MAK_RTIEXEC_HOSTNAME` in `/etc/hosts`. This requires a `link` statement when launching a container so that an entry is written into `/etc/hosts`. (See Section B.2.9 for more information.)

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

Note that containers of image `ma-lrc-base` are not started by themselves. Either a derived image (such as `ma-shipsim`) is started or `ma-lrc-base` is started in conjunction with a secondary container (such as `shipsim`).

B.2.13.8 `t1;dr`

```
docker run -d --name maklm --mac-address 02:C9:96:5B:12:B7 msaas-registry.cloudapp.net:5000/msaas/ma-lm
```

```
docker run -d --name rtiexec --link maklm:maklm msaas-registry.cloudapp.net:5000/msaas/mak-rtiexec
```

```
docker run -d --name shipsim --link rtiexec:rtiexec msaas-registry.cloudapp.net:5000/msaas/ma-shipsim
```

```
docker run -d --name shipsim2 --link rtiexec:rtiexec msaas-registry.cloudapp.net:5000/msaas/ma-shipsim -subscribe
```

Running these commands should result in a two federate federation with `shipsim` publishing a single surface vessel and `shipsim2` reflecting those values.

B.2.14 MÄK `rtiexec` Image (AUS, DST Group)

B.2.14.1 Image

```
app-docker136.hex.tno.nl:443/vtmak/mak-rtiexec
```

B.2.14.2 Description

This image provides an executable container for starting the MÄK RTI `rtiexec` process. The `rtiexec` connects to an RTI forwarder, which will be started automatically within the same container. This image derives from the `mak-rti-base` image which provides a default RID file.

The environment variables, listed in Table B-6, exist in an `rtiexec` container:

Table B-6: Environment Variables.

Environment Variable	Description	Default Value	Image
<code>MAK_LOG_LEVEL</code>	The level of detail provided in console/log output. Can be a value in the range 0 – 4 with 0 being no detail and 4 the most detail.	2	<code>mak-rtiexec</code>
<code>MAK_LOGFILE_DIR</code>	The directory into which an <code>rtiexec</code> log file will be written.	Working directory	<code>mak-rtiexec</code>
<code>MAK_RTIEXEC_LOGFILE</code>	The name of the log file written by the <code>rtiexec</code> . This needs to be defined for a log file to be written (into <code>MAK_LOGFILE_DIR</code>).	Not defined	<code>mak-rtiexec</code>

Environment Variable	Description	Default Value	Image
MAK_RTI_RID_FILE	Can be used to override the RID file used by the <code>rtiexec</code>	Not defined	<code>mak-rtiexec</code>
MAKLMGRD_LICENSE_FILE	The hostname (container name) of the MÄK License Manager server	@maklm	<code>mak-rti-base</code>
RTI_RID_FILE	The name of the RID file to use	/etc/makrti/rid.mtl	<code>mak-rti-base</code>
RTI_ASSISTANT_DISABLE	Disable the RTI Assistant that normally starts when the <code>rtiexec</code> and each federate start	1	<code>mak-rti-base</code>
MAK_RTIDIR	The installation directory of the MÄK RTI	/usr/local/makRti4.4.2	<code>mak-rti</code>

B.2.14.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/vtmak/mak-rtiexec
```

B.2.14.4 Docker Options

```
--name <container name>
```

The name of the `rtiexec` container will be used by federate containers to find the `rtiexec`. Explicitly specifying the name of the container makes this easier, although, the automatically assigned container name could be used.

```
--link <internal license manager name>:<license manager container name>
```

The `rtiexec` needs to connect to the MÄK Licence Manager. It does this by connecting to the server name specified by the value stored in the `MAKLMGRD_LICENSE_SERVER` environment variable. The default value, as set in the `mak-rti-base` image, for this environment variable is `@maklm`. Thus, the simplest way to specify this mapping, with a License Manager container name of `lm`, is `--link maklm:lm`.

B.2.14.5 Container Options

Nil.

B.2.14.6 Other Information

N/A.

B.2.14.7 Example

```
docker run -d --name rtiexec --link maklm:maklm app-docker136.hex.tno.nl:443/vtmak/mak-rtiexec
```

This assumes that the MÄK Licence Manager has been started in a container named maklm. If the Licence Manager is started with container name licman, use the following:

```
docker run -d --name rtiexec --link maklm:licman app-
docker136.hex.tno.nl:443/vtmak/mak-rtiexec
```

Both these examples rely on the value of the MAKLMGRD_LICENSE_SERVER environment variable being set to @maklm internal to the mak-rti-base image (from which, mak-rtiexec derives). If, for some reason, you want to change this, do the following (note that the --link is still required)

```
docker run -d --name rtiexec --link licman:maklm -e
"MAKLMGRD_LICENSE_SERVER=@licman" app-docker136.hex.tno.nl:443/vtmak/mak-
rtiexec
```

Again, this example assumes that the container running the License Manager is named maklm ... it is aliased within the rtiexec container to be licman.

B.2.15 MÄK VR Forces Image (TNO)

B.2.15.1 Image

```
app-docker136.hex.tno.nl:443/msaas-nld/vrf:pi
```

This image is based on LRC version 2.

B.2.15.2 Description

VR Forces (VRF) is a CGF from MaK that can be configured with many different simulation scenarios. This image version has a pre-configured demo scenario, but other VRF scenarios can be mounted via the docker volumes option. The scenario will start as soon as the container image is started.

By default, a demo scenario is started in wall clock time, using the FOM RPR_FOM_v2.0_1516-2010.xml. The scenario to be used and the way time is managed can be controlled via the container commandline options.

This image is currently only built against the Pitch LRC base image and requires a user defined MAC address to run (for more information, check image provider).

B.2.15.3 Synopsis

```
docker run <docker options> msaas-nld/vrf:pi <container options>
```

B.2.15.4 Docker Options

--mac-address=<MAC address> : MAC address for license purposes. **Required.**

-h mak : set the hostname to mak. **Required.**

-e DISPLAY=<xserver>:<displaynr> : set address of X-Server and display number to use. While nothing is actually going to be displayed, VRF requires a valid setting in order to run. **Required.**

B.2.15.5 Container Options

-v : verbose mode (default: no verbose mode).

-F <federation name> : join this federation (default: TheWorld).

-f <federate name> : use this federate name (default: VRF).

-s <scenario> : name of VRF scenario to use (default scenario for wall clock time). The corresponding scenario for logical time is called demo.logicaltime.

-t : start the VRF scenario in logical time and use TSO FOM RPR_FOM_v2.0_1516-2010-TSO.xml (default: start in wall clock time and use RO FOM RPR_FOM_v2.0_1516-2010.xml). When using logical time, the advancement of simulation time can be controlled by an external time pacer (see Pacer).

B.2.15.6 Other Information

N/A.

B.2.15.7 Example

The following Docker Compose file shows how to start VRF with a scenario called demo in wall clock time.

```
version: '2'

services:
  crc:
    image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
    mac_address: 00:18:8B:0D:4F:0B
    environment:
      - DISPLAY=xserver:0

  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver
    ports:
      - "8080:8080"

  kml:
    image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:pi
    command: -f KMLServer

  ge:
    image: app-docker136.hex.tno.nl:443/library/ge
    command: -f -r 10 Gulf http://kml:8080/kmlserver/entities
    environment:
      - DISPLAY=xserver:0

  vrf:
    image: app-docker136.hex.tno.nl:443/msaas-nld/vrf:pi
    mac_address: 8C:70:5A:0B:58:7E
    hostname: mak
    environment:
      - DISPLAY=xserver:0
```

B.2.16 Pacer Image (TNO)

B.2.16.1 Image

app-docker136.hex.tno.nl:443/msaas-nld/pacer:pi

app-docker136.hex.tno.nl:443/msaas-nld/pacer:po

app-docker136.hex.tno.nl:443/msaas-nld/pacer:none

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

The Pitch (pi) and Portico (po) images are based on LRC version 2. The none image (none) can be mounted as a volume to either one of the LRC base images.

B.2.16.2 Description

The Pacer is an application that can be used to control the advancement of simulation time in a time managed federation. The Pacer is controlled via a webbrowser providing buttons to:

- Pause/resume the simulation;
- Increase the simulation rate;
- Decrease the simulation rate; and
- Run the simulation as fast as possible.

The simulation rate is defined as $\text{simulation time} / \text{wallclock time}$. The rate = 1 corresponds to real-time.

The Pacer should join the federation as the first federate in order to control time advancement from the start. The Pacer starts in pause mode (time advanced state), that is, there is no time advancement. The resume button should be pressed in the webbrowser to start advancement.

B.2.16.3 Synopsis

```
docker run <docker options> msaas-nld/pacer:pi <container options>
```

```
docker run <docker options> msaas-nld/pacer:po <container options>
```

B.2.16.4 Docker Options

```
-e ADVERTISED_ADDRESS=<address>:<portnumber> : Optional (default: 127.0.0.1:8080)
```

The pacer will advertise its service on the given advertised address.

B.2.16.5 Container Options

```
-h, --help : Optional (no help)
```

```
-F, --federation <federation name> : Optional ($FEDERATIONNAME)
```

```
-f, --federate <federate name> : Optional (-)
```

```
-L, --logfile <log filename> : Optional (-)
```

```
-l, --loglevel <log level> : Optional (SEVERE)
```

```
-c, --configfile <configuration filename> : Optional (default.config)
```

```
-p, --port <port> : Optional (0)
```

With the `-p` option the application is instructed to open this port once it has created/joined the federation execution; can be used for bootstrapping (see LRC base image). If port number is zero then no port will be opened.

B.2.16.6 Other Information

B.2.16.6.1 Federation Object Models Modules

The Pacer uses the following FOM modules depending on configuration:

- default.config (default): Empty.xml
- alltsofoms.config: RPR_FOM_v2.0_1516-2010-TSO.xml, Sensor.xml, Damage.xml
- allrofroms.config: RPR_FOM_v2.0_1516-2010.xml, Sensor.xml, Damage.xml

B.2.16.6.2 Pacer Home Page

The home page of the pacer is http://<ADVERTISED_ADDRESS>/pacer/index.html.

B.2.16.6.3 Proxy

All URLs for the Pacer have as base URL `pacer`. A proxy can redirect all pacer requests based on this base URL.

B.2.16.6.4 Time Advancement

The Pacer is a time-stepped federate. The step size and look ahead is (in this implementation) fixed to 1. By increasing or decreasing the rate, the wall clock time between time advance requests is decreased or increased. Future versions of the Pacer may provide additional time control options.

B.2.16.7 Example

The following Docker Compose file defines a two-federate federation. The Pacer joins as the first federate and advertises its service on localhost, port 8081.

For Portico the first federate (the Pacer in this case) needs to join with all FOM modules that will be used in the federation (this is due to a FOM merging issue); the `allrofroms.config` or `alltsofoms.config` configuration of the Pacer needs to be used.

The `-p` option of the Pacer is used to signal other applications that the Pacer is ready for service. The LRC of DamageServer waits for this port to be opened before starting the federate application, i.e., DamageServer. This is indicated with the `LRC_MASTERADDRESS` environment variable.

```
version: '2'

services:
  pacer:
    image: app-docker136.hex.tno.nl:443/msaas-nld/pacer:po
    command: -f Pacer -c alltsofoms.config -p 6666
    environment:
      - ADVERTISED_ADDRESS=10.10.10.11:8081
    ports:
      - "8081:8080"

  damsims:
    image: app-docker136.hex.tno.nl:443/msaas-nld/damageserver:po
    command: -f DamageSim -c portico-logicaltime.config
    environment:
      - LRC_MASTERADDRESS=pacer:6666
```

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

Open a webbrowser and navigate to the address `http://10.10.10.11:8081/pacer/index.html`. The web page of the Pacer should appear. **Replace 10.10.10.11 by your host address.**

B.2.17 PDA Image (IFAD)

B.2.17.1 Image

`app-docker136.hex.tno.nl:443/msaas-dnk/pda:0.2`

B.2.17.2 Description

The IFAD PDA (Pitch DIS Adapter) image provides the *Pitch DIS Adapter Professional* as a service. The default config settings are for the IFAD Logger Service to record via the HLA standard (connection 'crc:8989' and federation TheWorld) and to please the dockerized version of IFAD ISIM Monotor (DIS Port 3200).

The image contains a VNC server for accessing the Pitch DIS Adapter GUI.

The PDA is MAC-address-licensed, therefore a running instance of the image needs a specific MAC address.

B.2.17.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/msaas-dnk/pda:0.2<
```

B.2.17.4 Docker Options

Recommended:

`-d` : Run container in background

`--name <container name>` : Assign a container name for simple startup of Pitch DIS Adapter service.

Required:

`--mac-address="00:19:9B:75:9F:7C"` : MAC address for license purposes

`--link <Pitch RTI host name: crc`

Optional:

`-p <port1>:5900` : VNC port number

B.2.17.5 Container Options

Nil.

B.2.17.6 Other Information

None.

B.2.17.7 Example

To start a container with the name `pda` and published port 5900 for the VNC server, use

```
docker run -d -p 5900:5900 --mac-address="00:19:9B:75:9F:7C" --link crc:crc --  
name pda app-docker136.hex.tno.nl:443/msaas-dnk/pda:0.2
```


To use the Pitch DIS Adapter GUI, connect via a VNC client on port 5900 and run the following from the provided terminal

```
./usr/local/PitchDISAdapter/bin/pitchdisadapter
```

To use the default settings and run the Pitch DIS Adapter service use

```
docker exec -it pda bin/sh -c '/usr/local/PitchDISAdapter/bin/pitchdisadapter-cmd'
```

This will connect the Pitch DIS Adapter to the Pitch RTI host `crc` via the HLA Federation `TheWorld` using the DIS port 3200 and DIS Broadcast Address 255.255.255.255.

B.2.18 Pitch CRC Image (TNO)

B.2.18.1 Image

```
app-docker136.hex.tno.nl:443/pitch/crc:5.3.0.0L
```

```
app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
```

B.2.18.2 Description

This image is the Pitch CRC (Central RTI Component). The CRC is licensed against a certain MAC address; hence the container must have this mac-address.

B.2.18.3 Synopsis

```
docker run <docker options> pitch/crc:<tag> <container options>
```

B.2.18.4 Docker Options

`--mac-address=00:18:8B:0D:4F:0B` : MAC address for license purposes. **Required.**

`-v <my settings file>:/root/prtl1516e/prtl1516eCRC.settings` : use the given settings file as base rather than the default settings file. **Optional.**

`-e CRC_NICKNAME=<crc name>` : Use this as the nickname for the CRC. Default is `crc-<container hostname>`. **Optional**

`-e CRC_PORT=<crc port>` : Use this CRC port number. Default is 8989. **Optional.**

`-e CRC_SKIP_CONNECTIVITY_CHECK=<boolean>` : Skip connectivity test back to connecting LRC. Default is `false`. **Optional.**

`-e CRC_REJECT_MISMATCHED_VERSIONS=<boolean>` : Reject LRCs with miss-matching version. Default is `true`. **Optional.**

`-e CRC_BOOSTERADDRESS=<booster host>:<booster port>` : Start in Booster Mode, using the given booster host address and port number. Default is Direct Mode. **Optional.**

`-e DISPLAY=<XServer host>:<Display number>` : if set, use this XServer display as GUI. For example, `xserver:0`. **Optional.**

If `CRC_BOOSTERADDRESS` and/or `DISPLAY` is set then the container will wait for the XServer and Booster to start. The number of wait attempts is determined by the `-w` option.

B.2.18.5 Container Options

`-h` : Display help information. Optional.

`-l <key>` : run license activator with the given key.

`-i` : Set interactive mode. This option should be used in combination with the docker option `-i` in order to use the container TTY. Default is non-interactive. Optional.

`-w <attempts>` : Number of attempts to wait for XServer (if `DISPLAY` set) and Booster (if `CRC_BOOSTERADDRESS` set), before starting the CRC (non-negative integer, default is indefinite).

B.2.18.6 Other Information

None.

B.2.18.7 Example

B.2.18.7.1 Example A: Run CRC as a Daemon

Run the CRC as a daemon and give the container the name “crc”; other containers can use this name to link to:

```
docker run -d \
  --mac-address=00:18:8B:0D:4F:0B \
  --name crc app-docker136.hex.tno.nl:443/pitch/crc:5.3.0.0L
```

B.2.18.7.2 Example B: Run CRC with Interactive Command Line

Run the CRC in interactive mode to use the CRC command line prompt:

```
docker run -it \
  --rm \
  --mac-address=00:18:8B:0D:4F:0B \
  --name crc app-docker136.hex.tno.nl:443/pitch/crc:5.3.0.0L -i
```

B.2.18.7.3 Example C: Run CRC with a GUI

Run the CRC as in example A, but using a GUI:

Start XServer first:

```
docker run -d \
  -p 8080:8080 \
  --name xserver app-docker136.hex.tno.nl:443/library/xserver
```

Start CRC:

```
docker run -d \
  --mac-address=00:18:8B:0D:4F:0B \
```

```
--link xserver \  
-e DISPLAY=xserver:0 \  
--name crc app-docker136.hex.tno.nl:443/pitch/crc:5.3.0.0L
```

And now open the web page <http://<HOSTADDRESS>:8080/vnc.html>.

B.2.19 Pitch DIS Adapter Image (TNO)

B.2.19.1 Image

app-docker136.hex.tno.nl:443/pitch/dis:2.6.0L

This image is based on LRC version 2.

B.2.19.2 Description

This image is the Pitch DIS Adapter (a DIS-HLA GW). The DIS Adapter is licensed against a certain mac-address, hence the container must have this MAC address. The Pitch DIS Adapter GUI can be accessed via an X Server.

B.2.19.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/pitch/dis:2.6.0L  
<container options>
```

B.2.19.4 Docker Options

--mac-address=00:18:8B:0D:4F:1B : MAC address for license purposes. Required.

B.2.19.5 Environment Variables

-e DISPLAY=<x display> : Address of X Display. Required.

B.2.19.6 Container Options

See also Pitch DIS Adapter manual.

-auto : Starts the DIS Adapter in autoconnect mode.

-config <file> : Give the config file to use in the DIS Adapter. NOTE that the config file must be mounted into the container to become accessible to the application.

B.2.19.7 Other Information

None.

B.2.19.8 Example

Start the composition below named pi-run.yml with the command `docker-compose -f pi-run.yml up`

```
version: '2'  
  
services:  
  dis:  
    image: app-docker136.hex.tno.nl:443/pitch/dis:2.6.0L  
    command: -auto
```

```

mac_address: 00:18:8B:0D:4F:1B
environment:
- DISPLAY=xserver:0
- MINSLEEP=5

crc:
image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.0.0L
mac_address: 00:18:8B:0D:4F:0B

web:
image: app-docker136.hex.tno.nl:443/pitch/web:2.1.1
ports:
- "8080:8080"

shipsim:
image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
command: -f ShipSim --fom-modules-dir allrofoms

xserver:
image: app-docker136.hex.tno.nl:443/library/xserver
ports:
- "8081:8080"

```

Notes:

- The Pitch DIS Adapter connects in this example automatically to the federation execution, hence the CRC is started in the composition as well.
- A MINSLEEP sleep period is used to allow the X Server to start before the DIS Adapter. A better practice is to have the X Server always running.
- The GUI of the DIS Adapter can be accessed via the X Server: <http://<HOSTADDRESS>:8080/vnc.html>.
- The DIS Adapter starts with a default configuration; settings can be changed via the GUI or via the `config` option.

B.2.20 Pitch Recorder Image (TNO)

B.2.20.1 Image

`app-docker136.hex.tno.nl:443/pitch/rec:2.2.0L`

This image is based on LRC version 2.

B.2.20.2 Description

This image is the Pitch Recorder for the recording and replay of simulation data. The Pitch Recorder GUI can be accessed with a XServer.

B.2.20.3 Synopsis

```
docker run <docker options> pitch/rec:2.2.0L <container options>
```

B.2.20.4 Docker Options

`--link <crc-name>` : Reference to name of the CRC container. Default name is `crc`. Optional.

`--mac-address=D4:BE:D9:26:AD:EB` : MAC address for license purposes. **Required**.

`-e DISPLAY=<XServer display>` : Address of X Display. **Required**, unless Pitch Recorder uses the `nogui` option.

B.2.20.5 Container Options

The container options are passed on to Pitch Recorder unchanged and have the following syntax:

```
<Recorder options> [ <project file> [ <project file options> ]]
```

B.2.20.5.1 Project File Options

Project File Options are:

- p, --play : Start Playback of the loaded project
- l, --loop : Start Loop Playback of the loaded project
- r, --rec : Start Recording of the loaded project
- crc <host[:port]> : Override CRC host in project file
- lsd <designator> : Override Local Settings Designator in project file
- federation <name> : Override federation name in project file

B.2.20.5.2 Recorder Options

Recorder Options are:

- n, --nogui : Do not display GUI
- c, --config <file> : Load config from (default is recorder.config)
- s, --system Look and Feel : Use System Look and Feel (default on Windows OS)
- j, --java Look and Feel : Use Java Look and Feel (default on non Windows OS)
- d, --debug : Show debug info
- h, --help : Display help info
- v, --version : Display version info

B.2.20.6 Other Information

None.

B.2.20.7 Example

B.2.20.7.1 Start Pitch Recorder with Docker Run Commands

B.2.20.7.1.1 Start XServer

```
docker run -d \  
-p 8080:8080 \  
--name xserver app-docker136.hex.tno.nl:443/library/xserver
```

B.2.20.7.1.2 Start CRC

```
docker run -d \
  --mac-address=00:18:8B:0D:4F:0B \
  --name crc app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
```

B.2.20.7.1.3 Start Recorder

```
docker run -d \
  --link xserver \
  --link crc \
  -e DISPLAY=xserver:0 \
  --mac-address=D4:BE:D9:26:AD:EB \
  --name rec app-docker136.hex.tno.nl:443/pitch/rec:2.2.0L
```

The Pitch Recorder will not connect automatically to a federation execution and is therefore not visible in the Pitch Webviewer. The Recorder needs to be configured and connected to a federation via its GUI.

B.2.20.7.1.4 Create a Data Image from a Recording

This example explains how to create a container data image from a data recording.

B.2.20.7.1.5 Create Directory Structure for Recordings

In this example Pitch Recorder will be configured to store the recorded data on the host filesystem rather than in the container. The following directory structure needs to be created in the directory where the compositions used in this example are started from:

```
mkdir -p ./recordings/projects
mkdir -p ./recordings/foms
mkdir -p ./recordings/databases
```

Copy the FOMs that you need for the recording into the `foms` directory.

B.2.20.7.1.6 Create a Recorder Project

Start Pitch Recorder to create a recorder project. Also, the CRC is started for project configuration reasons.

Start the composition: `docker-compose -f create-project.yml up`

Where the composition file is:

```
version: '2'

services:
  rec:
    image: app-docker136.hex.tno.nl:443/pitch/rec:2.2.0L
    mac_address: D4:BE:D9:26:AD:EB
    volumes:
      - ./recordings/projects:/usr/local/PitchRecorder/projects
      - ./recordings/foms:/usr/local/PitchRecorder/foms
      - ./recordings/databases:/usr/local/PitchRecorder/databases
    environment:
      - DISPLAY=xserver:0
```

```
crc:
  image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
  mac_address: 00:18:8B:0D:4F:0B
  environment:
    - CRC_REJECT_MISMATCHED_VERSIONS=false
    - DISPLAY=xserver:0

xserver:
  image: app-docker136.hex.tno.nl:443/library/xserver
  ports:
    - "8080:8080"
```

In Pitch Recorder do the following:

- Create a new project called `test`
- Add a datastream for HLA FOM Data:
 - Set the datastream name to `hlastream`
 - Set the Federation Name to `TheWorld`
 - Set the Pitch CRC Host to `crc` and
 - HLA FOM data as needed for the recording from the foms directory
 - Press OK and Apply; the datastream should show connected (this is why the CRC needs to be running)
 - Press OK to leave panel
- Add a channel for the datastream from Channel > New:
 - Set the channel name to `hlachannel` and press OK (i.e., everything will be recorded)
- Make sure that the `hlachannel` is on Rec
- Save the project in the `recordings > projects` directory, and name the file `test.recorder`

Terminate the composition with: `docker-compose -f create-project.yml down`.

B.2.20.7.1.7 Record Data

In this step a recording composition with Pitch Recorder is started. When the composition is started Pitch Recorder automatically starts recording. Note that the composition file assumes that the recorder project is saved as `test.recorder`.

Start the recording composition: `docker-compose -f record-data.yml up`.

Where the composition file is:

```
version: '2'

services:
  rec:
    image: app-docker136.hex.tno.nl:443/pitch/rec:2.2.0L
    mac_address: D4:BE:D9:26:AD:EB
    command: ./projects/test.recorder -r
    volumes:
```

```
- ./recordings/projects:/usr/local/PitchRecorder/projects
- ./recordings/foms:/usr/local/PitchRecorder/foms
- ./recordings/databases:/usr/local/PitchRecorder/databases
environment:
- DISPLAY=xserver:0

crc:
  image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
  mac_address: 00:18:8B:0D:4F:0B
  environment:
  - CRC_REJECT_MISMATCHED_VERSIONS=false
  - DISPLAY=xserver:0

xserver:
  image: app-docker136.hex.tno.nl:443/library/xserver
  ports:
  - "8080:8080"

shipsim:
  image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
  command: --fom-modules-dir allrofoms --scalable
```

Go to the Pitch Recorder window to see the progress. After some time stop the recording.

The project needs to be saved again before terminating Pitch Recorder. Before you save, select the Play setting for the `h1channel`. Save the project under `recordings > projects` as `test.recorder`.

Terminate the composition: `docker-compose -f record-data.yml down`.

Note that step 2 can also be done manually without a composition file, i.e., by manually opening the project and starting the recording.

B.2.20.7.1.8 Build Data Image

In this step a data image is created from the recording just done.

Do this with: `docker-compose -f build-dataimage.yml build`

Where the composition file is:

```
version: '2'

services:
  recordings:
    build:
      context: .
      dockerfile: Dockerfile
      image: app-docker136.hex.tno.nl:443/my/recording
```

And the Dockerfile:

```
FROM alpine:3.3

#copy data
COPY ./recordings/projects /usr/local/PitchRecorder/projects/
COPY ./recordings/foms /usr/local/PitchRecorder/foms/
```



```
COPY ./recordings/databases /usr/local/PitchRecorder/databases/

#Make application available as a volume
VOLUME ["/usr/local/PitchRecorder/projects"]
VOLUME ["/usr/local/PitchRecorder/foms"]
VOLUME ["/usr/local/PitchRecorder/databases"]

#Dummy command
CMD ["/bin/true"]
```

Note that in this example the data image is called `app-docker136.hex.tno.nl:443/my/recording`. For the creation of the data container image we use a small base image (Alpine Linux in this case).

B.2.20.7.1.9 Play Data

The data from the data image can be played with: `docker-compose -f play-data.yml up`

Where the composition file is:

```
version: '2'

services:
  rec:
    image: app-docker136.hex.tno.nl:443/pitch/rec:2.2.0L
    mac_address: D4:BE:D9:26:AD:EB
    command: ./projects/test.recorder -p
    volumes_from:
      - data
    environment:
      - DISPLAY=xserver:0

  data:
    image: app-docker136.hex.tno.nl:443/my/recording

  crc:
    image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
    mac_address: 00:18:8B:0D:4F:0B
    environment:
      - CRC_REJECT_MISMATCHED_VERSIONS=false
      - DISPLAY=xserver:0

  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver
    ports:
      - "8080:8080"
```

Go to the Pitch Recorder window to see progress.

B.2.20.7.1.10 Remove Recordings

Recordings can be removed from the databases and projects directories if they are not needed anymore.

B.2.21 Pitch WebGUI Image (TNO)

B.2.21.1 Image

app-docker136.hex.tno.nl:443/pitch/web:2.1.1

B.2.21.2 Description

This image is the Pitch Web GUI. The Web GUI can be used to monitor and control CRCs, federations and federates.

B.2.21.3 Synopsis

```
docker run <docker options> pitch/web:2.2.1 <container options>
```

B.2.21.4 Docker Options

--link <crc-name>[:<crc-name-alias>] : Reference to the name of the CRC container. Default name is crc. Optional.

-p 8080 : Web GUI port. **Required.**

B.2.21.5 Container Options

-h : displays help information. Optional.

-v : displays verbose output. Optional.

B.2.21.6 Webview Session Information

http://<docker-host>:<port>/Pitchwebview

Login: Administrator

Password: admin

B.2.21.7 Example

Run the Webviewer as a daemon and link to the CRC container.

```
docker run -d -p 8080:8080 --link crc:crc --name web app-docker136.hex.tno.nl:443/pitch/web:2.1.1
```

In a webbrowser open the Webviewer page:

http://<docker-host>:8080/Pitchwebview

B.2.22 Proxy Image (TNO)

B.2.22.1 Image

app-docker136.hex.tno.nl:443/library/proxy

B.2.22.2 Description

This image is a simple nginx based proxy that directs data from an external port to an internal service port. This can be useful in a multi-node cluster where it is not known where the service will be scheduled and a proxy can provide a fixed address and port to external clients.

Note that this proxy is most useful when running services in a cluster with Docker Swarm standalone. A proxy is not needed when using docker in Docker Swarm mode. In Swarm mode a service is accessible via its external port from every node.

B.2.22.3 Synopsis

```
docker run <docker options> library/proxy <container options>
```

B.2.22.4 Docker Options

-p <external port>:<internal port>:port specifications.

B.2.22.5 Container Options

```
{ [<internal port>]:<service name>:<internal service port>[:ws] }*
```

If the internal port is not provided then it defaults to the internal service port.

If ws is provided then the connection is for a websocket.

Multiple service mappings can be provided in the command. Only http is supported.

B.2.22.6 Other Information

N/A.

B.2.22.7 Example

A simple example where the xserver (internal service port 8080) is serviced via a proxy on port 8082.

```
version: '2'

services:
  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver

  proxy:
    image: app-docker136.hex.tno.nl:443/library/proxy
    command: 8082:xserver:8080:ws
    ports:
      - "8082:8082"
```

B.2.23 Sensor Server Image (TNO)

B.2.23.1 Image

app-docker136.hex.tno.nl:443/msaas-nld/sensorserver:pi

app-docker136.hex.tno.nl:443/msaas-nld/sensorserver:po

app-docker136.hex.tno.nl:443/msaas-nld/sensorserver:none

The Pitch (pi) and Portico (po) images are based on LRC version 2. The none image (none) can be mounted as a volume to either one of the LRC base images.

B.2.23.2 Description

This Sensor Server creates a cookie cutter Sensor based on the JROADS simulation environment. The Federate subscribes to Physical Entities from the RPR FOM and produces AbsoluteTracks for these entities as described in the Sensor FOM module.

The sensor may either be connected to an existing PhysicalEntity based on the HLA Object Instance name or placed at a static position. The update rate, range, azimuth and elevation are configurable using the properties file described below.

The sensor federate will always run in time stepped mode, but can be configured to be either (HLA) timemanaged or to run in (scaled) realtime.

B.2.23.3 Synopsis

```
docker run <docker options> msaas-nld/sensorserver:pi <container options>
```

```
docker run <docker options> msaas-nld/sensorserver:po <container options>
```

B.2.23.4 Docker Options

-h, --help: Optional (no help)

-p, --platform: Optional (HLA Object Instance name of the host platform)

-F, --federation <federation name>: Optional (\$FEDERATIONNAME)

-f, --federate <federate name>: Optional (-)

-c, --connectattempts <number>: Optional (\$LRC_CONNECTATTEMPTS)

-d, --directory <fom directory>: Optional (foms)

The RPR-FOM in the foms directory is **ReceiveOrder** (all interaction classes and object attributes are marked RO); alternatively use the directory tsofoms for **Timestamp Order** interaction classes and object attributes.

-timemanaged: Optional (false), run in time managed mode (both constrained and regulating)

-simtimestep <timestep in seconds>: Optional (1.0), in time managed mode, use this simulation time step

-realtimestep <timestep in seconds>: Optional (1.0), wallclock time step for dead reckoning of RO spatial

-L, --logfile <logfile name>: Optional (-)

-l, --loglevel <loglevel>: Optional (SEVERE)

B.2.23.5 Sensor Properties File

The properties of the sensor are configured in the `sensorfederate.properties` file that is provided with the image (at `root/application/sensorfederate.properties`). The properties file below is the standard file that is provided. It contains all available properties and a corresponding description.

```
# Sensor options

# Adds the Sensor to a dummy platform at specified [Lat Lon Alt] in [Degrees Meters]
latLonAlt = 51; 12; 1000

# Adds the sensor to the entity corresponding to the provided objectinstance name
# platform = HLA-Object-Instance-Name

# When a host platform is set this parameter can be used to configure the relative position [Forward
# Lateral Upward]
# specified in [Meters Meters] of the sensor with respect to the host platform
# relativePosition = 0 0 0

# Sets the update rate for the sensor, expressed in simulation time
updaterate = 2.5

# Sets the sensor range (from, to) in meters
rangeFrom = 0
rangeTo = 250000

# Sets the sensor elevation angle range (from, to) in degrees
elevationFrom = -90
elevationTo = 90

# Sets the sensor azimuth angle range (from, to) in degrees
azimuthFrom = -180
azimuthTo = 180
```

B.2.23.6 Used FOM Modules

The Sensor federate makes use of a custom Sensor FOM module (<https://github.com/globalsim/msaas-A/tree/master/FOMfiles/Sensor>). The Sensor FOM module describes AbsoluteTrack objects that are published by the Sensor federate.

The Sensor federate listens for published PhysicalEntities as defined by the RPR FOM (<https://github.com/globalsim/msaas-A/tree/master/FOMfiles/RPR>).

B.2.23.7 How to Run the Sensor Server

You can configure a docker-compose file to run the Federate in a small scenario. The example below starts the Ship Simulator, a Sensor Server, and a KML Server. The `iconurl` needs to be set to the host address of the KML Server, i.e., replace 10.10.10.11 by your host address.

All federates in this example use the Portico RTI. The example also shows how to configure the sensor with a properties file using volumes; the properties file is in this example located in the directory where docker-compose is started from.

```
version: '2'

services:
  sensorserver:
    image: app-docker136.hex.tno.nl:443/msaas-nld/sensorserver:po
    command: -f SensorServer
    volumes:
      - ./sensorfederate.properties:/root/application/sensorfederate.properties
```

```
kml:
  image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:po
  command: -f KMLServer -iconurl http://10.10.10.11:8090
  ports:
    - "8090:8080"
  environment:
    - LRC_MINSLEEP=5

shipsim:
  image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:po
  command: -f ShipSim
  environment:
    - LRC_MINSLEEP=5
```

B.2.24 SES Gulf (Service) Image (CPA ReDev GmbH)

B.2.24.1 Image

app-docker136.hex.tno.nl:443/msaas-ger/ses:gulf

B.2.24.2 Description

The SES-Service now consists of three services. The first one is an OGC-Web Feature Service (WFS) and a SESTextureService for providing the connected imagery. The SES image always depends on an image of the SESDatabase. There is no standalone usage possible since a database is required.

B.2.24.3 Synopsis

Checkout the example docker-compose.yml and move to that folder with Powershell (or your preferred terminal). Then type:

```
docker-compose up <docker-compose up options>
```

In an isolated environment, it will work as is. Otherwise, check the ports, which are defined in the yml-File.

B.2.24.4 Docker Compose Options

-d : Detached mode: getting back to terminal input while the services are running

B.2.24.5 Container Options

N/A.

B.2.24.6 Other Information

The initial start takes a little while. After downloading the images, the contained dump of the database content has to be restored into the DBMS. The imagery is stored with the service, as we wanted to have it as simple as possible. Otherwise we would have to add chargeable software.

B.2.24.7 Example

```
version: '3'
services:
  sesdb:
    restart: always
    image: app-docker136.hex.tno.nl:443/msaas-ger/sesdatabase:gulf
    #command: postgres -c config_file=/etc/postgresql.conf
    ports:
      - "5432:5432"
    environment:
```

```
- POSTGRES_PASSWORD=postgres
volumes:
- sesdata:/var/lib/postgresql/data

ses:
  image: app-docker136.hex.tno.nl:443/msaas-ger/ses:gulf
  ports:
  - "8080:8080"
  environment:
  - JAVA_OPTIONS=-Xmx2048m
  depends_on:
  - sesdb

volumes:
  sesdata:
    external: false
```

B.2.25 SES Meppen (Service) Image (CPA ReDev GmbH)

B.2.25.1 Image

app-docker136.hex.tno.nl:443/msaas-ger/ses:meppen

B.2.25.2 Description

The SES-Service yet consists of two services. The first one is an OGC-Web Feature Service (WFS) and a SESTextureService for providing the connected imagery. The SES image always depends on an image of the SESDatabase. There is no standalone usage possible since a database is required.

B.2.25.3 Synopsis

Checkout the example docker-compose.yml and move to that folder with Powershell (or your preferred terminal). Then type:

```
docker-compose up <docker-compose up options>
```

In an isolated environment it will work as is. Otherwise check the ports, which are defined in the yml-File.

B.2.25.4 Docker Compose Options

-d : Detached mode: getting back to terminal input while the services are running

B.2.25.5 Container Options

N/A.

B.2.25.6 Other Information

The initial start takes a while. After downloading the images, the contained dump of the database content has to be restored into the DBMS. To check the progress, it is helpful to display the console output. As the data folder of the DBMS is mapped to a docker volume, the dump must be read only once.

B.2.25.7 Example

```
version: '3'
services:
  sesdb:
    image: app-docker136.hex.tno.nl:443/msaas-ger/sesdatabase:meppen
    ports:
```

```
- "5432:5432"
environment:
  - POSTGRES_PASSWORD=postgres
volumes:
  - sesdata:/var/lib/postgresql/data

ses:
  image: app-docker136.hex.tno.nl:443/msaas-ger/ses
  ports:
    - "8080:8080"
  environment:
    - JAVA_OPTIONS=-Xmx2048m
  depends_on:
    - sesdb

volumes:
  sesdata:
    external: false
```

B.2.26 SESDatabase Gulf Image (CPA ReDev GmbH)

B.2.26.1 Image

app-docker136.hex.tno.nl:443/msaas-ger/sesdatabase:gulf

B.2.26.2 Description

The images of this kind contain data to be provided by the Synthetic Environment Service (SES). The tag describes the content of the included PostGIS-Database. As the imagery of the Gulf-Region-SES is delivered with the service, you cannot use this image standalone. It only contains the database part of an SES. You have to combine this database images with an image of the SES-Service. The compose file is located here.

B.2.26.3 Synopsis

`docker run app-docker136.hex.tno.nl:443/msaas-ger/sesdatabase:gulf`

B.2.26.4 Docker Options

`-p 5432:5432` : socket port number for remote access. Optional.

`-e POSTGRES_PASSWORD=postgres` : set the postgres password.

B.2.26.5 Container Options

N/A.

B.2.26.6 Other Information

None.

B.2.26.7 Example

None.

B.2.27 SESDatabase Meppen Image (CPA ReDev GmbH)

B.2.27.1 Image

`app-docker136.hex.tno.nl:443/msaas-ger/sesdatabase:meppen`

B.2.27.2 Description

The images of this kind contain data to be provided by the Synthetic Environment Service (SES). The tag describes the content of the included PostGIS-Database. It is not recommended – but possible – to use this image standalone. It only contains the database part of an SES. The recommended usage is to combine this database images with an image of the SES-Service. An example for the Meppen-Database is stored in the code section.

B.2.27.3 Synopsis

`docker run app-docker136.hex.tno.nl:443/msaas-ger/sesdatabase:meppen`

B.2.27.4 Docker Options

`-p 5432:5432` : socket port number for remote access. Optional.

`-e POSTGRES_PASSWORD=postgres` : set the postgres password.

`-v sesdata:/var/lib/postgresql/data` : define a volume for the storage of the data. (The volume must be created manually)

B.2.27.5 Container Options

N/A.

B.2.27.6 Other Information

None.

B.2.27.7 Example

None.

B.2.28 ShipSim Image (AUS, DST Group)

B.2.28.1 Image

`app-docker136.hex.tno.nl/msaas-aus/shipsim:none`

`app-docker136.hex.tno.nl/msaas-aus/shipsim:pi`

`app-docker136.hex.tno.nl/msaas-aus/shipsim:po`

The Pitch (pi) and Portico (po) images are based on LRC version 2. The none image (none) can be mounted as a volume to either one of the LRC base images.

B.2.28.2 Description

Contains a federate that simulates one or more ships. There are three behaviours for the generated ships:

- 1) A single ship will sail a straight line from an initial latitude and longitude with a given speed and course. This is the default.
- 2) A single ship will sail a set of waypoints in a loop.
- 3) Multiple ships will sail a set of waypoints in a loop with each ship having a random starting position along the set of waypoints.

See below for more information on these options.

B.2.28.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl/msaas-aus/shipsim:pi  
<container options>
```

B.2.28.4 Docker Options

`-p <host port>:8088`: The container starts a http server on port 8088 for reading ship data.

B.2.28.5 Container Options*B.2.28.5.1 Federate Options*

`-F, -federation-name`: The name of the federation to create/join (default is 'Federation').

`-f, -federate-name`: The name of the federate joining. Defaults to Platform. This is actually provided as the `FederateType`, rather than `FederateName`. The RTI is relied upon to provide a unique federate name, which is important when scaling multiple instances of the ShipSim federate.

`-r, -realtime-rate`: The number of one second platform state steps to take per one second of wallclock time (updates to the federation occur once per wallclock second), i.e., a value of `n` will make the ship move at `n` times realtime (approximately). (default is '2').

`-timemanaged`: Enable the use of HLA time management. The federate will be both time constrained and time regulating. Default is to not be time managed.

`-timestep`: When time managed, the value supplied to this option will be used as the time step for time advance requests. Defaults to 0.5.

`-lookahead`: The lookahead to use when the federate is time managed. Defaults to 0.1.

`-connection-attempts`: The number of times to attempt to connect to the CRC before quitting. This is necessary in contexts where the start of federation components is automated and the federate may begin before the CRC has fully started and begun accepting connections (default is '5').

`-subscribe`: An instance of ShipSim started with this option will not create any entities. Instead, it will subscribe to `SurfaceVessel` object instances and print to console all `reflectAttributeValues` callbacks received.

`-c, -crc-host-name`: The name/ip address of the host running the CRC (default is 'crc').

`-p, -crc-port`: The port on which the crc is listening (default is '8989').

`--scalable`: This option will have ShipSim allow the RTI to generate object instance names when registering platforms in the federation. These names are guaranteed to be unique and, so, allows a ShipSim federate to be scaled up using docker-compose commands. The default operation of ShipSim (when not specifying `--scalable`) is to use the marking (see `-marking` below) as the object instance name. Trying to use docker-compose to scale in this configuration will result in an attempt to register multiple object instances with the same name, which is not allowed by the RTI.

`--fom-modules <fom module 1> <fom module 2> ...`: allows you to specify FOM modules on the command line.

`--fom-modules-dir <dir>`: lets you specify a directory and ShipSim will load all files in that directory as FOM modules. The ShipSim image ships with the `allrofoms` and `alltsofoms` directories. The `allrofoms` directory contains the RO versions of the RPR, Damage, Empty and Sensor FOMs. The `alltsofoms` directory contains the TSO versions of the same FOMs.

Both options `--fom-modules` and `--fom-modules-dir` can be used together. If neither are used, ShipSim loads the `RPR_FOM_v2.0_1516-2010.xml` FOM.

B.2.28.5.2 Model Options

`-e, -entity-type`: The value of the EntityType attribute when creating a single ship (either straight running or waypoint following) (default is '1:3:13:6:1:1:0'). Note the use of colons to separate values ... this is required.

`-E, -entity-identifier`: The value of the EntityIdentifier attribute when creating a single ship (either straight running or waypoint following) (default is '1:1:1'). Note the use of colons to separate values ... this is required. The centre value is the Application ID and will be overwritten with the federate handle obtained by the joined federate to ensure the whole identifier is unique across all federates.

`-m, -marking`: The value of the Marking attribute when creating a single ship (either straight running or waypoint following) (default is 'Ship'). *This value is used as the object instance name of the registered SurfaceVessel and, therefore, needs to be unique within the federation.*

`-latitude`: Initial latitude in decimal degrees when creating a single straight running ship (default is '26.573106').

`-longitude`: Initial longitude in decimal degrees when creating a single straight running ship (default is '56.590209').

`-speed`: Initial speed in m/s when creating a single straight running ship (default is '10.0').

`-course`: Initial course in degrees from north when creating a single straight running ship (default is '270.0').

`-path-file <waypoint path filename>`: Specify a file that contains a waypoint path for a single created ship to follow. Supplying a waypoint path file results in any values supplied to `-latitude`, `-longitude`, `-speed` and `-course` to be ignored.

The waypoint is defined by three whitespace separated values: latitude (decimal degrees), longitude (decimal degrees), speed (m/s). One waypoint is defined per line. When the ship reaches the final waypoint, it will loop back to the first waypoint. The `msaas/shipsim` image contains two waypoint files in its working directory: `hormuz.txt` and `shortpath.txt`. `hormuz.txt` is a long loop from the Gulf of Oman through

the Strait of Hormuz into the Persian Gulf and back again, transiting the defined shipping lanes for that area. `shortpath.txt` is a square path approximately 30 nautical miles (0.5 degrees) per side. `hormuz.txt` is copied below for information:

```
25.612728 57.034723 10.0
26.512129 56.779510 10.0
26.673069 56.542583 10.0
26.577908 56.261343 10.0
26.369344 55.303031 10.0
26.372698 54.392558 10.0
26.173276 54.381525 10.0
26.176957 55.332729 10.0
26.439587 56.331878 10.0
26.456717 56.510735 10.0
26.392737 56.591607 10.0
25.700720 56.871214 10.0
```

`-background <background configuration filename>`: Specify a file that contains parameter values used to generate a large number of ships with random initial conditions and motions. Specifying a background configuration file causes any values supplied to `-path-file`, `-latitude`, `-longitude`, `-speed`, `-course`, `-entity-type`, `-entity-identifier`, and `-marking` to be ignored.

An example background configuration file is below. This file is included in the `msaas-aus/shipsim` image as `background.json`.

```
{
  numberOfShips: 100,
  countries: [13, 16, 45, 57, 100, 101, 102, 121, 153, 178, 189, 223, 224, 225],
  categories: {
    80: [1, 2, 3, 4, 5],
    81: [1, 2, 3, 4, 5, 6, 7],
    82: [1, 2, 3, 4, 5, 6, 7, 8],
    83: [1, 2, 3, 4, 5],
    84: [1, 2, 3, 4],
    85: [1, 2, 3, 4],
    86: [1, 2, 3, 4, 5]
  },
  paths: [
    "hormuz.txt"
  ]
}
```

The `numberOfShips` key specifies the number of ships to create in the simulation (this can be overridden with the `-background-count` option described below). The `countries` key provides a list of country codes. The `categories` key provides a mapping from category codes to lists of subcategory codes valid for the respective category code. A random value is chosen from each of these lists and used in the `EntityType` attribute for created ships. Finally, the `paths` key maps to a list of waypoint path filenames (as described above). Again, each ship created is assigned a random value from this list.

`-background-count <n>`: Override the value for `numberOfShips` sourced from the background configuration file.

B.2.28.6 HTTP Server

The federate runs a HTTP server on port 8088 (internal to the container). Navigating to the mapped port (also 8088 in the above command line) in a browser will show the ship attribute values being updated to the federation. (TODO: let the values be set via the browser at runtime).

B.2.28.7 Example

B.2.28.7.1 *Straight Running Single Ship*

```
docker run -d --name ship -p 8088:8088 \
  app-docker136.hex.tno.nl/msaas-aus/shipsim:pi
-F TheWorld -f Ship
```

B.2.28.7.2 *Waypoint Path Following Single Ship*

```
docker run -d --name ship -p 8088:8088
  app-docker136.hex.tno.nl/msaas-aus/shipsim:pi
-F TheWorld -f Ship -path-file hormuz.txt
```

B.2.28.7.3 *200 Random Ships*

```
docker run -d --name ship -p 8088:8088
  app-docker136.hex.tno.nl/msaas-aus/shipsim:pi
-F TheWorld -f Ship -background background.json -background-count 200
```

B.2.29 ShipUI Image (AUS, DST Group)

B.2.29.1 Image

app-docker136.hex.tno.nl:443/msaas-aus/shipui:none

app-docker136.hex.tno.nl:443/msaas-aus/shipui:pi

app-docker136.hex.tno.nl:443/msaas-aus/shipui:po

The Pitch (pi) and Portico (po) images are based on LRC version 2. The none image (none) can be mounted as a volume to either one of the LRC base images.

B.2.29.2 Description

Contains a federate that provides an interface for visualising ship state and requesting weapon fire against other ships.

The interface is presented as a web based API and an HTML front end. The front end is accessible at localhost:8089 and makes use of the Web API that is available as subpaths below localhost:8089/api. The Web API is as follows, in Table B-7:

Table B-7: Web API.

Endpoint	Method	Argument	Return	Description
http://localhost:8089/api/ships	GET	Nil	{ships: [Ship1, Ship2, ...]}	Returns a list of ships known to the ShipUI federate.

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

Endpoint	Method	Argument	Return	Description
<code>http://localhost:8089/api/fire</code>	POST	<code>{source: Ship1, target: Ship2}</code>	Nil	Request a WeaponFire interaction be sent. A WeaponFire will only be sent if the source and target are known to the ShipUI federate and are different.

B.2.29.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/msaas-aus/shipui:pi
<container options>
```

B.2.29.4 Docker Options

`-p <host port>:8089`: The container starts a http server on port 8089 for reading ship data.

TIP: The user interface is stored in `/root/application/ui`. Using a volume mount to a host location allows for the UI to be changed and played with in realtime.

B.2.29.5 Container Options

`-F, --federation-name`: Set the name of the federation to create/join. Defaults to “Federation”.

`-f, --federate-name`: Set the name to use for the federate when joining the federation. Defaults to “ShipUI”.

`--fom-modules`: Specify a list of FOM modules for this federate to use.

`--fom-modules-dir`: Specify a directory that contains FOM modules for this federate to use.

`--timemanaged`: Run this federate in time managed mode, both time constrained and time regulating. If this option is not specified, the federate runs at realtime from the system clock.

`--timestep`: Specify the timestep for this federate to use when running time managed. Defaults to 0.5.

`--lookahead`: Specify the lookahead for this federate to use when running time managed. Defaults to 0.1.

B.2.29.6 Other Information

None.

B.2.29.7 Example

None.

B.2.30 Start Image (TNO)

B.2.30.1 Image

app-docker136.hex.tno.nl:443/msaas-nld/start:none

LRC Version 1:

app-docker136.hex.tno.nl:443/msaas-nld/start:pi

app-docker136.hex.tno.nl:443/msaas-nld/start:pi-centos

app-docker136.hex.tno.nl:443/msaas-nld/start:pi-debian

app-docker136.hex.tno.nl:443/msaas-nld/start:po

app-docker136.hex.tno.nl:443/msaas-nld/start:po-debian

app-docker136.hex.tno.nl:443/msaas-nld/start:ma

LRC Version 2:

app-docker136.hex.tno.nl:443/msaas-nld/start:pi-alpine-2

app-docker136.hex.tno.nl:443/msaas-nld/start:pi-centos-2

app-docker136.hex.tno.nl:443/msaas-nld/start:pi-debian-2

app-docker136.hex.tno.nl:443/msaas-nld/start:po-alpine-2

app-docker136.hex.tno.nl:443/msaas-nld/start:po-debian-2

B.2.30.2 Description

This is a simple federate, called start, that (optionally) provides an X Window GUI in which it displays all of the joined federates. Images (pi, po, ma) are provided for three RTIs, and one image (none) that can be mounted as a volume to either of the three LRC base images.

B.2.30.3 Synopsis

```
docker run <docker options> msaas-nld/start:<tag> <container options>
```

B.2.30.4 Docker Options

-e DISPLAY=<host address>:<display number> : use this X display. Optional.

If DISPLAY is invalid (e.g., the X Server is not up) then the application continues without X display.

-v ./foms:/root/application/foms : create/join federation with the FOM modules under the directory foms. Optional.

B.2.30.5 Container Options

-h, --help : Optional (no help)

-F, --federation <federation name> : Optional (\$FEDERATIONNAME)

-f, --federate <federate name> : Optional (-)

`-L, --logfile <log filename>: Optional (-)`

`-l, --loglevel <log level>: Optional (SEVERE)`

`-c, --connectattempts <connect attempts>: Optional ($LRC_CONNECTATTEMPTS).`

`-d, --directory <FOM directory>: Optional (foms)`

With the `-d` option the application can be started with different FOMs, i.e., the application creates/joins the federation execution with the following FOMs:

- `foms (default) : Empty.xml`
- `allfoms : Empty.xml, Damage.xml, Sensor.xml, RPR_FOM_v2.0_1516-2010-TSO.xml`
- `allrofoms : Empty.xml, Damage.xml, Sensor.xml, RPR_FOM_v2.0_1516-2010.xml`

`-p, --port <port>: Optional (0)`

With the `-p` option the application is instructed to open this port once it has created/joined the federation execution; can be used for bootstrapping (see LRC base image). If port number is zero then no port will be opened.

B.2.30.6 Other Information

None.

B.2.30.7 Example

For an X Server example, see X Server.

In the following example one master and three slaves are started. The slaves will wait for the master to create/join the federation execution.

```
version: '2'

services:
  master:
    image: app-docker136.hex.tno.nl:443/msaas-nld/start:po
    command: -p 8990

  start1:
    image: app-docker136.hex.tno.nl:443/msaas-nld/start:po
    environment:
      - LRC_MASTERHOST=master
      - LRC_MASTERPORT=8990

  start2:
    image: app-docker136.hex.tno.nl:443/msaas-nld/start:po
    environment:
      - LRC_MASTERHOST=master
      - LRC_MASTERPORT=8990

  start3:
    image: app-docker136.hex.tno.nl:443/msaas-nld/start:po
    environment:
      - LRC_MASTERHOST=master
      - LRC_MASTERPORT=8990
```


The output is something like:

```

Creating examples_start3_1
Creating examples_start1_1
Creating examples_start2_1
Creating examples_master_1
Attaching to examples_start3_1, examples_start1_1, examples_start2_1, examples_master_1
start3_1 | LRC: Version: portico/lrc:nightly-20160528-alpine-v0.3
start3_1 | LRC: Process settings: /root/lrc/settings.sh
start3_1 | LRC: Wait for master at master:8990
start1_1 | LRC: Version: portico/lrc:nightly-20160528-alpine-v0.3
start1_1 | LRC: Process settings: /root/lrc/settings.sh
start1_1 | LRC: Wait for master at master:8990
start2_1 | LRC: Version: portico/lrc:nightly-20160528-alpine-v0.3
start2_1 | LRC: Process settings: /root/lrc/settings.sh
start2_1 | LRC: Wait for master at master:8990
master_1 | LRC: Version: portico/lrc:nightly-20160528-alpine-v0.3
master_1 | LRC: Process settings: /root/lrc/settings.sh
master_1 | LRC: No sleep period
master_1 | LRC: Start: /root/application/start.sh
master_1 | 1::20:16:00.520::federate.Main::main::federationName = TheWorld
master_1 | 1::20:16:00.522::federate.Main::main::federateName = null
master_1 | 1::20:16:00.522::federate.Main::main::connectAttempts = 0
master_1 | 1::20:16:00.522::federate.Main::main::fomDirectory = foms
master_1 | 1::20:16:00.522::federate.Main::main::logFile = null
master_1 | 1::20:16:00.522::federate.Main::main::logLevel = INFO
master_1 | 1::20:16:00.522::federate.Main::main::port = 8,990
master_1 | 1::20:16:00.523::federate.Federate::connect::Attempt #1 of INFINITE to connect ...
master_1 | 1::20:16:00.523::federate.Federate::connect::Connected to RTI
master_1 | 1::20:16:00.523::federate.Federate::getFOMs::FOM: found 1 FOMs
master_1 | 1::20:16:00.524::federate.Federate::getFOMs::FOM[1]: file:/root/application/foms/Empty.xml
master_1 | -----
master_1 | GMS: address=a14d7bc8e60c-12629, cluster=TheWorld, physical address=172.21.0.5:42527
master_1 | -----
start3_1 | LRC: Wait for master at master:8990
start1_1 | LRC: Wait for master at master:8990
start2_1 | LRC: Wait for master at master:8990
start3_1 | LRC: Wait for master at master:8990
start1_1 | LRC: Wait for master at master:8990
start2_1 | LRC: Wait for master at master:8990
start3_1 | LRC: Wait for master at master:8990
start1_1 | LRC: Wait for master at master:8990
start2_1 | LRC: Wait for master at master:8990
master_1 | 1::20:16:04.848::federate.Federate::join::Created Federation
master_1 | 1::20:16:04.862::federate.Federate::join::Joined Federation
master_1 | 1::20:16:04.865::federate.Federate::publishAndSubscribe::Published and Subscribed
master_1 | 1::20:16:04.868::federate.Federate::start::Start ...
master_1 | 1::20:16:04.868::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128960331)
master_1 | 22::20:16:04.871::federate.Main$1::run::Waiting for clients on port: 8990
start3_1 | LRC: Wait for master at master:8990
master_1 | 22::20:16:05.054::federate.Main$1::run::Knock knock from: 172.21.0.3
start1_1 | LRC: Wait for master at master:8990
master_1 | 22::20:16:05.133::federate.Main$1::run::Knock knock from: 172.21.0.2
start2_1 | LRC: Wait for master at master:8990
master_1 | 22::20:16:05.205::federate.Main$1::run::Knock knock from: 172.21.0.4
start3_1 | LRC: Master master:8990 is up
start3_1 | LRC: No sleep period
start3_1 | LRC: Start: /root/application/start.sh
start1_1 | LRC: Master master:8990 is up
start1_1 | LRC: No sleep period
start1_1 | LRC: Start: /root/application/start.sh
start2_1 | LRC: Master master:8990 is up
start2_1 | LRC: No sleep period
start2_1 | LRC: Start: /root/application/start.sh
start3_1 | 1::20:16:06.646::federate.Main::main::federationName = TheWorld
start3_1 | 1::20:16:06.647::federate.Main::main::federateName = null
start3_1 | 1::20:16:06.650::federate.Main::main::connectAttempts = 0

```

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

```

start3_1 | 1::20:16:06.651::federate.Main::main::fomDirectory = foms
start3_1 | 1::20:16:06.651::federate.Main::main::logFile = null
start3_1 | 1::20:16:06.652::federate.Main::main::logLevel = INFO
start3_1 | 1::20:16:06.653::federate.Main::main::port = 0
start3_1 | 1::20:16:06.654::federate.Federate::connect::Attempt #1 of INFINITE to connect ...
start3_1 | 1::20:16:06.654::federate.Federate::connect::Connected to RTI
start3_1 | 1::20:16:06.656::federate.Federate::getFOMs::FOM: found 1 FOMs
start3_1 | 1::20:16:06.658::federate.Federate::getFOMs::FOM[1]: file:/root/application/foms/Empty.xml
start1_1 | 1::20:16:06.755::federate.Main::main::federationName = TheWorld
start1_1 | 1::20:16:06.756::federate.Main::main::federateName = null
start1_1 | 1::20:16:06.756::federate.Main::main::connectAttempts = 0
start1_1 | 1::20:16:06.757::federate.Main::main::fomDirectory = foms
start1_1 | 1::20:16:06.760::federate.Main::main::logFile = null
start1_1 | 1::20:16:06.760::federate.Main::main::logLevel = INFO
start1_1 | 1::20:16:06.760::federate.Main::main::port = 0
start1_1 | 1::20:16:06.761::federate.Federate::connect::Attempt #1 of INFINITE to connect ...
start1_1 | 1::20:16:06.761::federate.Federate::connect::Connected to RTI
start1_1 | 1::20:16:06.761::federate.Federate::getFOMs::FOM: found 1 FOMs
start1_1 | 1::20:16:06.763::federate.Federate::getFOMs::FOM[1]: file:/root/application/foms/Empty.xml
start2_1 | 1::20:16:06.962::federate.Main::main::federationName = TheWorld
start2_1 | 1::20:16:06.966::federate.Main::main::federateName = null
start2_1 | 1::20:16:06.967::federate.Main::main::connectAttempts = 0
start2_1 | 1::20:16:06.969::federate.Main::main::fomDirectory = foms
start2_1 | 1::20:16:06.970::federate.Main::main::logFile = null
start2_1 | 1::20:16:06.970::federate.Main::main::logLevel = INFO
start2_1 | 1::20:16:06.971::federate.Main::main::port = 0
start2_1 | 1::20:16:06.972::federate.Federate::connect::Attempt #1 of INFINITE to connect ...
start2_1 | 1::20:16:06.974::federate.Federate::connect::Connected to RTI
start2_1 | 1::20:16:06.975::federate.Federate::getFOMs::FOM: found 1 FOMs
start2_1 | 1::20:16:06.980::federate.Federate::getFOMs::FOM[1]: file:/root/application/foms/Empty.xml
start3_1 | -----
start3_1 | GMS: address=65eb972c1c27-13467, cluster=TheWorld, physical address=172.21.0.3:57944
start3_1 | -----
start1_1 | -----
start1_1 | GMS: address=1f1a8cb4cf79-6660, cluster=TheWorld, physical address=172.21.0.2:41383
start1_1 | -----
start2_1 | -----
start2_1 | GMS: address=78ea1a7d9a40-32339, cluster=TheWorld, physical address=172.21.0.4:60833
start2_1 | -----
start3_1 | ERROR [main] portico.lrc.jgroups: FAILURE createFederation: already exists, name=TheWorld
start3_1 | 1::20:16:09.557::federate.Federate::join::Didn't create federation, it already existed
start1_1 | ERROR [main] portico.lrc.jgroups: FAILURE createFederation: already exists, name=TheWorld
start1_1 | 1::20:16:09.576::federate.Federate::join::Didn't create federation, it already existed
master_1 | 1::20:16:09.615::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966144)
master_1 | 1::20:16:09.642::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966247)
start3_1 | 1::20:16:09.649::federate.Federate::join::Joined Federation
start1_1 | 1::20:16:09.652::federate.Federate::join::Joined Federation
start3_1 | 1::20:16:09.652::federate.Federate::publishAndSubscribe::Published and Subscribed
start3_1 | 1::20:16:09.653::federate.Federate::start::Start ...
start3_1 | 1::20:16:09.653::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966144)
start3_1 | 1::20:16:09.654::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966247)
start3_1 | 1::20:16:09.654::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128960331)
start1_1 | 1::20:16:09.659::federate.Federate::publishAndSubscribe::Published and Subscribed
start1_1 | 1::20:16:09.659::federate.Federate::start::Start ...
start1_1 | 1::20:16:09.660::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966247)
start1_1 | 1::20:16:09.661::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966144)
start1_1 | 1::20:16:09.662::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128960331)
start2_1 | ERROR [main] portico.lrc.jgroups: FAILURE createFederation: already exists, name=TheWorld
start2_1 | 1::20:16:10.650::federate.Federate::join::Didn't create federation, it already existed

```

```
start1_1 | 1::20:16:10.702::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966340)
start3_1 | 1::20:16:10.703::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966340)
master_1 | 1::20:16:10.705::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966340)
start2_1 | 1::20:16:10.741::federate.Federate::join::Joined Federation
start2_1 | 1::20:16:10.746::federate.Federate::publishAndSubscribe::Published and Subscribed
start2_1 | 1::20:16:10.746::federate.Federate::start::Start ...
start2_1 | 1::20:16:10.747::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966340)
start2_1 | 1::20:16:10.747::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966144)
start2_1 | 1::20:16:10.748::federate.Federate::addFederate::Added Federate MOM.Federate(Federate-
1483128966247)
start2_1 | 1::20:16:10.748::federate.Federate::addFederate::Added Federate MOM.Federate(Federat
```

B.2.31 Symbol Service Image (IFAD)

B.2.31.1 Image

app-docker136.hex.tno.nl:443/msaas-dnk/disenumerationsymbolservice:2.2

B.2.31.2 Description

The service translates Distributed Interactive Simulation (DIS) enumeration values to APP6(B) symbol codes and unit symbols. It also translates to an APP6(C) 20 digit description.

B.2.31.3 Synopsis

```
docker run <docker options> app-docker136.hex.tno.nl:443/msaas-
dnk/disenumerationsymbolservice:2.2
```

B.2.31.4 Docker Options

Recommended:

-d: Run container in background

Required:

-p <host port>:80

B.2.31.5 Container Options

Nil.

B.2.31.6 Service Interface

- /<service>/<forceidentifier>/<entitytype>
- /<service>/<entitytype> (same as /<service>/0/<entitytype>)

Where,

- forceidentifier is 0, 1, 2 or 3.
- entitytype is a sequence of zero to seven dot separated digits.
- service is one of app6b-symbolcode, dis-kind, dis-domain, dis-country, dis-category, dis-subcategory, dis-extra, dis-xml, app6c-description, app6c-code20digit,

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

app6b-symbolpng, app6b-symbolpng<N>, app6b-symbolsvg and app6b-symbolsvgXXXX, where each X ranges in 0-9 and N is one of 0030, 0035, 0050, 0055, 0060, 0065, 0070 and 0100.

Further details once container is running (see Examples below) at: <http://<host>:<port>/about>.

B.2.31.7 Examples

Start the service on e.g., port 953 on localhost.

```
docker run -d -p 953:80 app-docker136.hex.tno.nl:443/msaas-  
dnk/disenumerationsymbolservice:2.2
```

A service manual is now found at

<http://localhost:953/about>

Take forceidentifier as 1 and entitytype as 1.2.78.1.6.1.0 then:

- <http://localhost:953/app6b-symbolcode/1/1.2.78.1.6.1.0> returns

SFAPMFF-----S

- <http://localhost:953/dis-kind/1/1.2.78.1.6.1.0> returns

Platform

- <http://localhost:953/dis-xml/1/1.2.78.1.6.1.0> returns

```
<dis>  
  <kind>Platform</kind>  
  <domain>Air</domain>  
  <country>Germany</country>  
  <category>Fighter/Air Defense</category>  
  <subcategory>Eurofighter</subcategory>  
  <specific>Eurofighter GS</specific>  
  <extra></extra>  
</dis>
```

- <http://localhost:953/app6b-symbolsvg/1/1.2.78.1.6.1.0> returns

```
<svg baseProfile="tiny" height="46.2" version="1.2" viewBox="41 26 122 132" width="42.7"  
xmlns="http://www.w3.org/2000/svg">  
  <path d="M 155,150 C 155,50 115,30 100,30 85,30 45,50 45,150" fill="rgb(128,224,255)" fill-opacity="1"  
stroke="black" stroke-width="4"></path>  
  <text fill="black" font-family="Arial" font-size="45" font-weight="bold" stroke="none" stroke-width="4"  
text-anchor="middle" x="100" y="115">F</text>  
</svg>
```

B.2.32 Syslog Image (TNO)

B.2.32.1 Image

app-docker136.hex.tno.nl:443/library/syslog

B.2.32.2 Description

This syslog image can be used as general logging service for container output. Container output can be directed to the syslog service by setting the container logging driver to syslog.

B.2.32.3 Synopsis

`docker run <docker options> library/syslog <container options>`

B.2.32.4 Docker Options

`-p <syslog port number>:514` (**Required** to use syslog)

B.2.32.5 Container Options

See *rsyslog* at <https://linux.die.net/man/8/rsyslogd>.

B.2.32.6 Other Information

N/A.

B.2.32.7 Example

Docker compose file `compose-syslog.yml` to start the syslog service:

```
version: '2'

services:
  log:
    image: app-docker136.hex.tno.nl:443/library/syslog
    container_name: syslog
    ports:
      - "5000:514"
```

Docker compose file `compose-sim.yml` to start a few simulation services:

```
version: '2'

# Example to demonstrate the use of the log driver.
# In this example we use the syslog driver, and configure it to use format rfc3164
# so that logstash can interpret to data correctly as well.
# For the syslog address 127.0.0.1 is used, which is interpreted by the docker daemon
# as the address of the local Docker Host. If syslog runs on another Docker Host,
# the address of that host should be used instead.

services:
  kml:
    image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:po
    command: -f KMLServer -iconurl http://127.0.0.1:8090 -l FINEST
    ports:
      - "8090:8080"
    logging:
      driver: syslog
      options:
        syslog-address: "tcp://127.0.0.1:5000"
        syslog-format: rfc3164
        tag: "kml"

  start:
    image: app-docker136.hex.tno.nl:443/msaas-nld/start:po
    command: -f Start -l FINEST
    logging:
      driver: syslog
      options:
        syslog-address: "tcp://127.0.0.1:5000"
        syslog-format: rfc3164
        tag: "start"
```

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

To run the example, do the following:

- Start syslog service: `docker-compose -f compose-syslog.yml up -d`
- Start simulation services: `docker-compose -f compose-sim.yml up -d`
- Look at the log: `docker exec syslog tail -f /var/log/messages`

B.2.33 X Server Image (TNO)

B.2.33.1 Image

`app-docker136.hex.tno.nl:443/library/xserver`

B.2.33.2 Description

This image is an X Server for use by any X11 based application. The user can connect to the X Server with a VNC Client or with any modern webbrowser that supports websockets. Simultaneous connections from multiple clients are supported. Optionally a password can be configured that the user must supply to connect.

B.2.33.3 Synopsis

```
docker run <docker options> library/xserver <container options>
```

B.2.33.4 Docker Options

```
-p <VNC port number>:5900 (optional)
-p <Websocket port number>:8080 (optional)
-p <Supervisor port number>:9001 (optional)
-e DISPLAY=<X11 DISPLAY value> (optional, default :0)
-e PASSWORD=<Password value> (optional, default no password)
-e DISPLAY_WIDTH=<display width> (optional, default 1024)
-e DISPLAY_HEIGHT=<display height> (optional, default 768)
```

B.2.33.5 Container Options

N/A.

B.2.33.6 Other Information

None.

B.2.33.7 Example

The following Docker Compose file provides an example with a Portico federate that links with the X Server. The helloworld service starts with a delay of MINSLEEP seconds in order to allow the X Server to start. The xserver service exposes port 8080 so that the user can connect to the X Server from a webbrowser. Browser URL is <http://<Docker Host address>:8080/vnc.html>.

```
version: '2'

services:
  helloworld:
    image: app-docker136.hex.tno.nl:443/msaas-nld/start:po
    environment:
      - DISPLAY=xserver:0
      - MINSLEEP=5

  xserver:
    image: app-docker136.hex.tno.nl:443/library/xserver
    ports:
      - "8080:8080"
```

To run these containers directly from the command line, do:

```
docker run -d --name xserver -p 8080:8080 app-
docker136.hex.tno.nl:443/library/xserver
```

```
docker run -d --link xserver -e DISPLAY=xserver:0 app-
docker136.hex.tno.nl:443/msaas-nld/start:po
```

B.3 FOMS

B.3.1 Sensor FOM

B.3.1.1 Module Overview

See Figure B-2 and Table B-8 for an overview and description of the module.

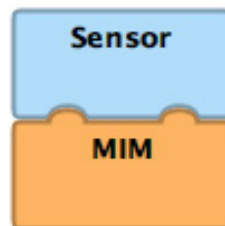


Figure B-2: Sensor Module Overview.

Table B-8: Module Description.

Module	Description
Sensor	Provides a simple FOM for the communication of sensor tracks.

B.3.1.2 Object Classes Overview

Figure B-3 provides an overview of the sensor object classes.



Figure B-3: Sensor Object Classes Overview.

B.3.1.3 Identification

Table B-9 lists sensor module identification.

Table B-9: Sensor Module Identification.

Name	Sensor Module
Type	FOM
Version	1.0
Modification Date	2016-07-05
Security Classification	Unclassified
Purpose	
Application Domain	Analysis
Description	Provides a simple FOM for the communication of sensor tracks.
Use Limitation	
Other	

B.3.1.4 Object Classes

Figure B-4 shows sensor object classes.



Figure B-4: Sensor Object Classes.

B.3.1.4.1 Track

B.3.1.4.1.1 HLAobjectRoot.Track

Table B-10 shows a base class used to represent tracks. These are principally those generated and output from sensors but tracks formed through a fusion of data may also be represented. Its attributes are those common to all tracks.

Table B-10: HLAobjectRoot.Track Attributes.

Attribute	Datatype	Semantics
TrackTime	HLAfloat64Time	Optional attribute (default: time of receipt). The time at which the current track attributes are associated with.
TrackTimeOfInitiation	HLAfloat64Time	Optional attribute (default: unknown). The time at which the track was initiated.

Attribute	Datatype	Semantics
TrackStatus	TrackStatusEnum	Optional attribute (default: DETECTED). Indicates the status of the track.
TrackDescription	HLAASCIIString	Optional attribute (default: none). A user defined description of the track.
TrackNumber	TrackNumber	Required attribute. The track number assigned by the sensor generating the track.
TargetObjectSetValid	HLABoolean	Optional attribute (default: false). If true, indicates that the value of the TargetObjectInstanceNameSet attribute is valid. If false, the value of the TargetObjectInstanceNameSet must be ignored.
TargetObjectSet	ObjectInstanceNameArray	Optional attribute (default: none). The object instance names of the objects that may give rise to this track. This can be used as a shortcut for track correlation or for visualisation purposes. Ideally, there is only one such object associated with the track. However, given that some sensors may not be able to resolve nearly collocated entities, there may be more than one object instance associated with it. The set may also be empty, indicating that no object gave rise to this track (a 'false' track). This attribute is optional if the TargetObjectInstanceNameSetValid attribute is false.
TrackGenerator	ObjectInstanceName	Optional attribute (default: none). The name of the object instance that generates the track. This will typically be a Sensor object instance.
TrackQuality	TrackQualityEnum	Optional attribute (default: TQ0). Provides a somewhat subjective – to the sensor generating the track – measure of the reliability of the track data. The number refers to a measure of uncertainty 1 (high uncertainty) to 15 (little uncertainty).
TrackClassification	TrackClassificationArray	Optional attribute (default: no classification). An estimate of the classification of the tracked object. This attribute does not include information about the intent of the tracked object, i.e., hostile, friendly etc. This attribute consists of a list of possible entity types along with a measure of the confidence in each possible value.
TrackIdentification	TrackIdentificationEnum	Optional attribute (default: PENDING). A classification of the intention of the object that is being tracked.

B.3.1.4.2 *AbsoluteTrack*

B.3.1.4.2.1 **HLAObjectRoot.Track.AbsoluteTrack**

This class represents tracks that are defined in terms of the earth centred, earth fixed coordinate system that defines the WGS84 ellipsoid (see Table B-11).

Table B-11: HLAObjectRoot.Track.AbsoluteTrack Attributes.

Attribute	Datatype	Semantics
PositionValid	HLABoolean	Optional attribute (default: false). A flag with value true if the Position attribute can be relied upon.
Position	TrackPositionStruct	Required attribute if PositionValid is true. An estimate of the position of the track.
PositionError	TrackPositionStruct	Required attribute if PositionValid is true. The error with which the position is given.
VelocityValid	HLABoolean	Optional attribute (default: false). A flag with value true if the Velocity attribute can be relied upon.
Velocity	TrackVelocityStruct	Required attribute if VelocityValid is true. An estimate of the velocity of the track.
VelocityError	TrackVelocityStruct	Required attribute if VelocityValid is true. The error with which the velocity is given.
<i>TrackTime</i>	HLAfloat64Time	Optional attribute (default: time of receipt). The time at which the current track attributes are associated with.
<i>TrackTimeOfInitiation</i>	HLAfloat64Time	Optional attribute (default: unknown). The time at which the track was initiated.
<i>TrackStatus</i>	TrackStatusEnum	Optional attribute (default: DETECTED). Indicates the status of the track.
<i>TrackDescription</i>	HLAASCIIString	Optional attribute (default: none). A user defined description of the track.
<i>TrackNumber</i>	TrackNumber	Required attribute. The track number assigned by the sensor generating the track.
<i>TargetObjectSetValid</i>	HLABoolean	Optional attribute (default: false). If true, indicates that the value of the TargetObjectInstanceNameSet attribute is valid. If false, the value of the TargetObjectInstanceNameSet must be ignored.

Attribute	Datatype	Semantics
<i>TargetObjectSet</i>	ObjectInstanceNameArray	<p>Optional attribute (default: none). The object instance names of the objects that may give rise to this track.</p> <p>This can be used as a shortcut for track correlation or for visualisation purposes. Ideally, there is only one such object associated with the track. However, given that some sensors may not be able to resolve nearly co-located entities, there may be more than one object instance associated with it. The set may also be empty, indicating that no object gave rise to this track (a ‘false’ track).</p> <p>This attribute is optional if the TargetObjectInstanceNameSetValid attribute is false.</p>
<i>TrackGenerator</i>	ObjectInstanceName	Optional attribute (default: none). The name of the object instance that generates the track. This will typically be a Sensor object instance.
<i>TrackQuality</i>	TrackQualityEnum	Optional attribute (default: TQ0). Provides a somewhat subjective – to the sensor generating the track – measure of the reliability of the track data. The number refers to a measure of uncertainty 1 (high uncertainty) to 15 (little uncertainty).
<i>TrackClassification</i>	TrackClassificationArray	Optional attribute (default: no classification). An estimate of the classification of the tracked object. This attribute does not include information about the intent of the tracked object, i.e., hostile, friendly etc. This attribute consists of a list of possible entity types along with a measure of the confidence in each possible value.
<i>TrackIdentification</i>	TrackIdentificationEnum	Optional attribute (default: PENDING). A classification of the intention of the object that is being tracked.

B.3.1.5 Data Types

B.3.1.5.1 Simple Data Types

Table B-12 shows simple data types.

Table B-12: Simple Data Types.

Name	Units	Semantics
TrackNumber	NA	Track number, a non-negative integer.

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

Name	Units	Semantics
TrackConfidenceMeasure	NA	Measure of confidence between 0.0 and 1.0.
TrackDistance	meter	Distance in meters.
TrackVelocity	meter/second	Velocity in m/s.
TrackCountryCode	NA	Country code in TrackEntityTypeStruct.

B.3.1.5.2 Enumerated Data Types

B.3.1.5.2.1 TrackQualityEnum

An enumeration of TrackQuality attribute values can be seen in Table B-13.

Representation: HLAinteger32BE.

Table B-13: TrackQualityEnum Enumerations.

Enumerator	Value
TQ0	0
TQ1	1
TQ2	2
TQ3	3
TQ4	4
TQ5	5
TQ6	6
TQ7	7
TQ8	8
TQ9	9
TQ10	10
TQ11	11
TQ12	12
TQ13	13
TQ14	14
TQ15	15

B.3.1.5.2.2 TrackIdentificationEnum

An enumeration that indicates the identification of a track, as shown in Table B-14.

- * **Pending** – A track which has not been subjected to the identification process.
- * **Unknown** – An evaluated track which has not been identified.
- * **Assumed Friend** – A track which is assumed to be a friend because of its characteristics, behavior, or origin.
- * **Friend** – A track belonging to a declared friendly nation.
- * **Neutral** – A track whose characteristics, behavior, origin, or nationality indicate that it is neither supporting nor opposing friendly forces
- * **Suspect** – A track which is potentially hostile because of its characteristics, behavior, origin or nationality.
- * **Hostile** – A track declared to belong to any opposing nation, party, group, or entity, which by virtue of its behavior or information collected on it such as characteristics, origin or nationality contributes to the threat to friendly forces.

Representation: HLAinteger32BE.

Table B-14: TrackIdentificationEnum Enumerations.

Enumerator	Value
PENDING	0
UNKNOWN	1
ASSUMED_FRIEND	2
FRIEND	3
NEUTRAL	4
SUSPECT	5
HOSTILE	6

B.3.1.5.2.3 TrackStatusEnum

An enumeration that indicates the status of the track, as shown in Table B-15. The sequence is DETECTED => CONFIRMATION => TRACKING => LOST.

- * **DETECTED**: initial state of track. The track is in this state for the first N detections of the RWO.
- * **CONFIRMATION**: next state of track. The track is in this state for the next M detections of the RWO.
- * **TRACKING**: The track enters this state once it is confirmed.
- * **LOST**: The track has been lost. A track that has been lost may still have value as it provides an indicator of an entity's presence and motion at some time in the past.

Representation: HLAinteger32BE.

Table B-15: TrackStatusEnum Enumerations.

Enumerator	Value
DETECTED	0
CONFIRMATION	1
TRACKING	2
LOST	3

B.3.1.5.3 Array Data Types

As shown in Table B-16.

Table B-16: Array Data Types.

Name	Element Datatype	Cardinality	Encoding	Semantics
ObjectInstanceName	HLAASCIIchar	Dynamic	HLAvariableArray	The instance name of an object registered with the RTI.
ObjectInstanceNameArray	ObjectInstanceName	Dynamic	HLAvariableArray	Array of ObjectInstance Name elements.
TrackClassificationArray	TrackClassificationStruct	Dynamic	HLAvariableArray	Array of Classification Struct elements.

B.3.1.5.4 Fixed Record Data Types

B.3.1.5.4.1 TrackClassificationStruct

Table B-17 defines a structure representing the potential entities being tracked as well as a measure of confidence in the platform classification.

Encoding: HLAfixedRecord.

Table B-17: TrackClassificationStruct Data Types.

Name	Datatype	Semantics
EntityType	TrackEntityTypeStruct	This field provides a possible identity for the platform being tracked.

Name	Datatype	Semantics
ConfidenceMeasure	TrackConfidenceMeasure	This field provides a measure, as a number between 0 and 1, of the degree of confidence in the platform classification.

B.3.1.5.4.2 TrackPositionStruct

The location of an object in the world coordinate system, as specified in IEEE Std 1278.1-1995 Section 1.3.2, as shown in Table B-18.

Encoding: HLAfixedRecord.

Table B-18: TrackPositionStruct Data Types.

Name	Datatype	Semantics
X	TrackDistance	Distance from the origin along the X axis.
Y	TrackDistance	Distance from the origin along the Y axis
Z	TrackDistance	Distance from the origin along the Z axis

B.3.1.5.4.3 TrackVelocityStruct

Table B-19 shows the rate at which the position is changing over time.

Encoding: HLAfixedRecord.

Table B-19: TrackVelocityStruct Data Types.

Name	Datatype	Semantics
XVelocity	TrackVelocity	Velocity component along the X axis.
YVelocity	TrackVelocity	Velocity component along the Y axis.
ZVelocity	TrackVelocity	Velocity component along the Z axis.

B.3.1.5.4.4 TrackEntityTypeStruct

Type of entity. Based on the Entity Type record as specified in IEEE 1278.1-1995 Section 5.2.16, as shown in Table B-20.

Encoding: HLAfixedRecord.

Table B-20: TrackEntityTypeStruct Data Types.

Name	Datatype	Semantics
EntityKind	HLAbyte	Kind of entity.
Domain	HLAbyte	Domain in which the entity operates.
CountryCode	TrackCountryCode	Country to which the design of the entity is attributed.
Category	HLAbyte	Main category that describes the entity.
Subcategory	HLAbyte	Subcategory to which an entity belongs based on the Category field.
Specific	HLAbyte	Specific information about an entity based on the Subcategory field.
Extra	HLAbyte	Extra information required to describe a particular entity.

B.3.2 Damage FOM

B.3.2.1 Module Overview

Figure B-5 and Table B-21 provide an overview of the damage module.

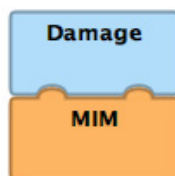


Figure B-5: Damage Module Overview.

Table B-21: Module Overview.

Module	Description
Damage	Provides a simple FOM for the communication of damage reports on entities in response to detonations.

B.3.2.2 Interaction Classes Overview

Figure B-6 provides an overview of the damage interaction classes.



Figure B-6: Damage Interaction Classes Overview.

B.3.2.3 Identification

Table B-22 outlines the damage module identification.

Table B-22: Damage Module Identification.

Name	Damage Module
Type	FOM
Version	1.0
Modification Date	2016-10-11
Security Classification	Unclassified
Purpose	
Application Domain	Analysis
Description	Provides a simple FOM for the communication of damage reports on entities in response to detonations.
Use Limitation	
Other	

B.3.2.4 Interaction Classes

Figure B-7 shows damage interaction classes.



Figure B-7: Damage Interaction Classes.

B.3.2.4.1 DamageReport

B.3.2.4.1.1 HLAinteractionRoot.DamageReport

This is the base class for damage reports.

B.3.2.4.2 EntityDamageReport

B.3.2.4.2.1 HLAinteractionRoot.DamageReport.EntityDamageReport

This class represents a damage report for an entity. It should only be sent for entities that are affected by a detonation and there can be multiple damage reports on a single detonation. Most of the parameters are

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

copied from the Munition Detonation interaction. Specific to the report are the DamageLocation, DamageState and TargetObjectIdentifier (see Table B-23).

Table B-23: HLAinteractionRoot.DamageReport.EntityDamageReport Parameters.

Parameter	Datatype	Semantics
DamageLocation	WorldLocationStruct	The location at which the damage is assessed.
DamageState	DamageStatusEnum32	Damage status.
FiringObjectIdentifier	RTIobjectId	The identifier of the object instance firing the munition, if any. From the MunitionDetonation.
TargetObjectIdentifier	RTIobjectId	The identifier of the object instance to which this damage report applies.
MunitionObjectIdentifier	RTIobjectId	The identifier of the munition, if any. From the MunitionDetonation.
MunitionType	EntityTypeStruct	Type of munition (if any), from MunitionDetonation.
WarheadType	WarheadTypeEnum16	Warhead type (if any), from MunitionDetonation.
EventIdentifier	EventIdentifierStruct	Identifier to track WeaponFire and MunitionDetonations (if any), from MunitionDetonation.

B.4 COMPOSITIONS

B.4.1 Composition A: Big Single Node

An example of a composition where all services run on a single node, connected via a bridge network. The only interface to the outside world is a X Server display accessible on host address port 8080.

The composition includes:

- EPIC as COP Viewer.
- Synthetic Environment Service.
- Several scenario generation services (VRF, Shipment).
- Infrastructure components (CRC, DIS/HLA GW, Google Earth, Google Chrome, XServer, IFAD Icon Server).

```
version: '2'

# Docker compose file where all components run on a single host

services:

# EPIC components
diswebgw:
  image: app-docker136.hex.tno.nl:443/msaas-usa/dis-web-gw:epic

epic:
  image: app-docker136.hex.tno.nl:443/epic/epic:cax
  environment:
    - dis_gw_url=diswebgw
    - ses_url=http://ses:8080/SgjWMS/WMS

# SES components
sesdb:
  image: app-docker136.hex.tno.nl:443/msaas-ger/sesdatabase:gulf
  environment:
    - POSTGRES_PASSWORD=postgres
  volumes:
    - sesdata:/var/lib/postgresql/data

ses:
  image: app-docker136.hex.tno.nl:443/msaas-ger/ses:gulf
  environment:
    - JAVA_OPTIONS=-Xmx2048m
  depends_on:
    - sesdb

# HLA components
crc:
  image: app-docker136.hex.tno.nl:443/pitch/crc:5.3.2.1L
  mac_address: 00:18:8B:0D:4F:0B
  environment:
    - DISPLAY=xserver:0

dis:
  image: app-docker136.hex.tno.nl:443/pitch/dis:2.6.0L
  command: -auto
  mac_address: 00:18:8B:0D:4F:1B
  environment:
    - DISPLAY=xserver:0

shipsim:
  image: app-docker136.hex.tno.nl:443/msaas-aus/shipsim:pi
  command: --fom-modules-dir allrofoms --scalable

kml:
  image: app-docker136.hex.tno.nl:443/msaas-nld/kmlserver:pi
  command: -f KMLServer -iconurl http://icon:80/app6b-symbolpng

vrf:
  image: app-docker136.hex.tno.nl:443/msaas-nld/vrf:pi
  mac_address: 8C:70:5A:0B:58:7E
  hostname: mak
  environment:
    - DISPLAY=xserver:0

# General components
icon:
  image: app-docker136.hex.tno.nl:443/msaas-dnk/disenumerationsymbolservice:2.2

ge:
  image: app-docker136.hex.tno.nl:443/library/ge
  command: -f -r 10 Gulf http://kml:8080/kmlserver/entities
  environment:
    - DISPLAY=xserver:0
```

ANNEX B – DOCKER CONTAINER IMAGE DESCRIPTIONS

```
gc:
  image: app-docker136.hex.tno.nl:443/library/gc
  command: --homepage http://epic:7070/epic
  environment:
    - DISPLAY=xserver:0

xserver:
  image: app-docker136.hex.tno.nl:443/library/xserver
  ports:
    - "8080:8080"

volumes:
  sesdata:
    external: false
```

REPORT DOCUMENTATION PAGE																			
1. Recipient's Reference	2. Originator's References STO-TR-MSG-136-Part-VII AC/323(MSG-136)TP/834	3. Further Reference ISBN 978-92-837-2160-4	4. Security Classification of Document PUBLIC RELEASE																
5. Originator Science and Technology Organization North Atlantic Treaty Organization BP 25, F-92201 Neuilly-sur-Seine Cedex, France																			
6. Title Modelling and Simulation as a Service, Volume 4: Experimentation Report																			
7. Presented at/Sponsored by Developed by NATO MSG-136.																			
8. Author(s)/Editor(s) Multiple			9. Date July 2019																
10. Author's/Editor's Address Multiple			11. Pages 188																
12. Distribution Statement There are no restrictions on the distribution of this document. Information about the availability of this and other STO unclassified publications is given on the back cover.																			
13. Keywords/Descriptors <table border="0"> <tr> <td>Cloud computing</td> <td>NATO C3 Classification Taxonomy</td> </tr> <tr> <td>Composability</td> <td>Reference architecture</td> </tr> <tr> <td>Distributed simulation</td> <td>Service-Oriented Architecture (SOA)</td> </tr> <tr> <td>Interoperability</td> <td>Simulation</td> </tr> <tr> <td>Live, Virtual, Constructive (LVC) Modelling</td> <td>Simulation Architecture</td> </tr> <tr> <td>Modelling and Simulation (M&S)</td> <td>Simulation Environments</td> </tr> <tr> <td>Modelling and Simulation as a Service (MSaaS)</td> <td>Simulation Interoperability</td> </tr> <tr> <td>M&S Services</td> <td></td> </tr> </table>				Cloud computing	NATO C3 Classification Taxonomy	Composability	Reference architecture	Distributed simulation	Service-Oriented Architecture (SOA)	Interoperability	Simulation	Live, Virtual, Constructive (LVC) Modelling	Simulation Architecture	Modelling and Simulation (M&S)	Simulation Environments	Modelling and Simulation as a Service (MSaaS)	Simulation Interoperability	M&S Services	
Cloud computing	NATO C3 Classification Taxonomy																		
Composability	Reference architecture																		
Distributed simulation	Service-Oriented Architecture (SOA)																		
Interoperability	Simulation																		
Live, Virtual, Constructive (LVC) Modelling	Simulation Architecture																		
Modelling and Simulation (M&S)	Simulation Environments																		
Modelling and Simulation as a Service (MSaaS)	Simulation Interoperability																		
M&S Services																			
14. Abstract <p>M&S as a Service (MSaaS) is a concept that combines service orientation and the provision of M&S applications via the as-a-service model of cloud computing to enable more composable simulation environments that can be deployed and executed on-demand. NATO MSG-136 investigated the concept of MSaaS and provided technical and organizational foundations to establish the Allied Framework for M&S as a Service within NATO and partner nations. The Allied Framework for M&S as a Service is the common approach of NATO and nations towards implementing MSaaS and is defined by the Operational Concept Document, Technical Reference Architecture, and MSaaS Governance Policies.</p> <p>This document describes the MSG-136 experimentation activities that evaluated the MSaaS concept in two experiments with eight test cases in total. The MSaaS Reference Architecture of MSG-136 proved to be valid; the technology MSG-136 used was well manageable. The experimentation results demonstrate that MSaaS is capable of realizing the vision that M&S products, data and processes are conveniently accessible to a large number of users whenever and wherever needed.</p>																			





BP 25

F-92201 NEUILLY-SUR-SEINE CEDEX • FRANCE
Télécopie 0(1)55.61.22.99 • E-mail mailbox@cs0.nato.int



DIFFUSION DES PUBLICATIONS STO NON CLASSIFIEES

Les publications de l'AGARD, de la RTO et de la STO peuvent parfois être obtenues auprès des centres nationaux de distribution indiqués ci-dessous. Si vous souhaitez recevoir toutes les publications de la STO, ou simplement celles qui concernent certains Panels, vous pouvez demander d'être inclus soit à titre personnel, soit au nom de votre organisation, sur la liste d'envoi.

Les publications de la STO, de la RTO et de l'AGARD sont également en vente auprès des agences de vente indiquées ci-dessous.

Les demandes de documents STO, RTO ou AGARD doivent comporter la dénomination « STO », « RTO » ou « AGARD » selon le cas, suivi du numéro de série. Des informations analogues, telles que le titre est la date de publication sont souhaitables.

Si vous souhaitez recevoir une notification électronique de la disponibilité des rapports de la STO au fur et à mesure de leur publication, vous pouvez consulter notre site Web (<http://www.sto.nato.int/>) et vous abonner à ce service.

CENTRES DE DIFFUSION NATIONAUX

ALLEMAGNE

Streitkräfteamt / Abteilung III
Fachinformationszentrum der Bundeswehr (FIZBw)
Gorch-Fock-Straße 7, D-53229 Bonn

BELGIQUE

Royal High Institute for Defence – KHID/IRSD/RHID
Management of Scientific & Technological Research
for Defence, National STO Coordinator
Royal Military Academy – Campus Renaissance
Renaissancelaan 30, 1000 Bruxelles

BULGARIE

Ministry of Defence
Defence Institute “Prof. Tsvetan Lazarov”
“Tsvetan Lazarov” bul no.2
1592 Sofia

CANADA

DGSIST 2
Recherche et développement pour la défense Canada
60 Moodie Drive (7N-1-F20)
Ottawa, Ontario K1A 0K2

DANEMARK

Danish Acquisition and Logistics Organization
(DALO)
Lautrupbjerg 1-5
2750 Ballerup

ESPAGNE

Área de Cooperación Internacional en I+D
SDGPLATIN (DGAM)
C/ Arturo Soria 289
28033 Madrid

ESTONIE

Estonian National Defence College
Centre for Applied Research
Riia str 12
Tartu 51013

ETATS-UNIS

Defense Technical Information Center
8725 John J. Kingman Road
Fort Belvoir, VA 22060-6218

FRANCE

O.N.E.R.A. (ISP)
29, Avenue de la Division Leclerc
BP 72
92322 Châtillon Cedex

GRECE (Correspondant)

Defence Industry & Research General
Directorate, Research Directorate
Fakinos Base Camp, S.T.G. 1020
Holargos, Athens

HONGRIE

Hungarian Ministry of Defence
Development and Logistics Agency
P.O.B. 25
H-1885 Budapest

ITALIE

Ten Col Renato NARO
Capo servizio Gestione della Conoscenza
F. Baracca Military Airport “Comparto A”
Via di Centocelle, 301
00175, Rome

LUXEMBOURG

Voir Belgique

NORVEGE

Norwegian Defence Research
Establishment
Attn: Biblioteket
P.O. Box 25
NO-2007 Kjeller

PAYS-BAS

Royal Netherlands Military
Academy Library
P.O. Box 90.002
4800 PA Breda

POLOGNE

Centralna Biblioteka Wojskowa
ul. Ostrobramska 109
04-041 Warszawa

PORTUGAL

Estado Maior da Força Aérea
SDFA – Centro de Documentação
Alfragide
P-2720 Amadora

REPUBLIQUE TCHEQUE

Vojenský technický ústav s.p.
CZ Distribution Information Centre
Mladoboleslavská 944
PO Box 18
197 06 Praha 9

ROUMANIE

Romanian National Distribution
Centre
Armaments Department
9-11, Drumul Taberei Street
Sector 6
061353 Bucharest

ROYAUME-UNI

Dstl Records Centre
Rm G02, ISAT F, Building 5
Dstl Porton Down
Salisbury SP4 0JQ

SLOVAQUIE

Akadémia ozbrojených síl gen.
M.R. Štefánika, Distribučné a
informačné stredisko STO
Demänová 393
031 06 Liptovský Mikuláš 6

SLOVENIE

Ministry of Defence
Central Registry for EU & NATO
Vojkova 55
1000 Ljubljana

TURQUIE

Milli Savunma Bakanlığı (MSB)
ARGE ve Teknoloji Dairesi
Başkanlığı
06650 Bakanlıklar – Ankara

AGENCES DE VENTE

**The British Library Document
Supply Centre**
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
ROYAUME-UNI

**Canada Institute for Scientific and
Technical Information (CISTI)**
National Research Council Acquisitions
Montreal Road, Building M-55
Ottawa, Ontario K1A 0S2
CANADA

Les demandes de documents STO, RTO ou AGARD doivent comporter la dénomination « STO », « RTO » ou « AGARD » selon le cas, suivie du numéro de série (par exemple AGARD-AG-315). Des informations analogues, telles que le titre et la date de publication sont souhaitables. Des références bibliographiques complètes ainsi que des résumés des publications STO, RTO et AGARD figurent dans le « NTIS Publications Database » (<http://www.ntis.gov>).



BP 25

F-92201 NEUILLY-SUR-SEINE CEDEX • FRANCE
Télécopie 0(1)55.61.22.99 • E-mail mailbox@cs.o.nato.int



DISTRIBUTION OF UNCLASSIFIED STO PUBLICATIONS

AGARD, RTO & STO publications are sometimes available from the National Distribution Centres listed below. If you wish to receive all STO reports, or just those relating to one or more specific STO Panels, they may be willing to include you (or your Organisation) in their distribution.

STO, RTO and AGARD reports may also be purchased from the Sales Agencies listed below.

Requests for STO, RTO or AGARD documents should include the word 'STO', 'RTO' or 'AGARD', as appropriate, followed by the serial number. Collateral information such as title and publication date is desirable.

If you wish to receive electronic notification of STO reports as they are published, please visit our website (<http://www.sto.nato.int/>) from where you can register for this service.

NATIONAL DISTRIBUTION CENTRES

BELGIUM

Royal High Institute for Defence –
KHID/IRSD/RHID
Management of Scientific & Technological
Research for Defence, National STO
Coordinator
Royal Military Academy – Campus
Renaissance
Renaissancelaan 30
1000 Brussels

BULGARIA

Ministry of Defence
Defence Institute "Prof. Tsvetan Lazarov"
"Tsvetan Lazarov" bul no.2
1592 Sofia

CANADA

DSTKIM 2
Defence Research and Development Canada
60 Moodie Drive (7N-1-F20)
Ottawa, Ontario K1A 0K2

CZECH REPUBLIC

Vojenský technický ústav s.p.
CZ Distribution Information Centre
Mladoboleslavská 944
PO Box 18
197 06 Praha 9

DENMARK

Danish Acquisition and Logistics Organization
(DALO)
Lautrupbjerg 1-5
2750 Ballerup

ESTONIA

Estonian National Defence College
Centre for Applied Research
Riia str 12
Tartu 51013

FRANCE

O.N.E.R.A. (ISP)
29, Avenue de la Division Leclerc – BP 72
92322 Châtillon Cedex

GERMANY

Streitkräfteamt / Abteilung III
Fachinformationszentrum der
Bundeswehr (FIZBw)
Gorch-Fock-Straße 7
D-53229 Bonn

GREECE (Point of Contact)

Defence Industry & Research General
Directorate, Research Directorate
Fakinos Base Camp, S.T.G. 1020
Holargos, Athens

HUNGARY

Hungarian Ministry of Defence
Development and Logistics Agency
P.O.B. 25
H-1885 Budapest

ITALY

Ten Col Renato NARO
Capo servizio Gestione della Conoscenza
F. Baracca Military Airport "Comparto A"
Via di Centocelle, 301
00175, Rome

LUXEMBOURG

See Belgium

NETHERLANDS

Royal Netherlands Military
Academy Library
P.O. Box 90.002
4800 PA Breda

NORWAY

Norwegian Defence Research
Establishment, Attn: Biblioteket
P.O. Box 25
NO-2007 Kjeller

POLAND

Centralna Biblioteka Wojskowa
ul. Ostrobramska 109
04-041 Warszawa

PORTUGAL

Estado Maior da Força Aérea
SDFA – Centro de Documentação
Alfragide
P-2720 Amadora

ROMANIA

Romanian National Distribution Centre
Armaments Department
9-11, Drumul Taberei Street
Sector 6
061353 Bucharest

SLOVAKIA

Akadémia ozbrojených síl gen
M.R. Štefánika, Distribučné a
informačné stredisko STO
Demänová 393
031 06 Liptovský Mikuláš 6

SLOVENIA

Ministry of Defence
Central Registry for EU & NATO
Vojkova 55
1000 Ljubljana

SPAIN

Área de Cooperación Internacional en I+D
SDGPLATIN (DGAM)
C/ Arturo Soria 289
28033 Madrid

TURKEY

Milli Savunma Bakanlığı (MSB)
ARGE ve Teknoloji Dairesi Başkanlığı
06650 Bakanlıklar – Ankara

UNITED KINGDOM

Dstl Records Centre
Rm G02, ISAT F, Building 5
Dstl Porton Down, Salisbury SP4 0JQ

UNITED STATES

Defense Technical Information Center
8725 John J. Kingman Road
Fort Belvoir, VA 22060-6218

SALES AGENCIES

The British Library Document Supply Centre

Boston Spa, Wetherby
West Yorkshire LS23 7BQ
UNITED KINGDOM

Canada Institute for Scientific and Technical Information (CISTI)

National Research Council Acquisitions
Montreal Road, Building M-55
Ottawa, Ontario K1A 0S2
CANADA

Requests for STO, RTO or AGARD documents should include the word 'STO', 'RTO' or 'AGARD', as appropriate, followed by the serial number (for example AGARD-AG-315). Collateral information such as title and publication date is desirable. Full bibliographical references and abstracts of STO, RTO and AGARD publications are given in "NTIS Publications Database" (<http://www.ntis.gov>).