Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Converting VMSA Federates through Polka 2.0

HLA 1.3 to IEEE 1516

Briand Gaudet
SG Software Solutions Incorporated

Prepared by:
SG Software Solutions Incorporated
75 Bristol Avenue
Stillwater Lake, Nova Scotia B3Z 1E9

PWGSC Contract No: W7707-078006/001/HAL
Contract Scientific Authority: Tania E. Randall, 902-426-3100 ext 283

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2008-020
June 2008

Canada

This page intentionally left blank.

Converting VMSA Federates through Polka 2.0

HLA 1.3 to IEEE 1516

Briand Gaudet
SG Software Solutions Incorporated

Prepared by:
SG Software Solutions Incorporated
75 Bristol Avenue
Stillwater Lake, Nova Scotia B3Z 1E9

Contract number: W7707-078006/001/HAL
Contract Scientific Authority: Tania E. Randall, 902-426-3100 ext 283

Defence R&D Canada – Atlantic

Contract Report

DRDC Atlantic CR 2008-020

June 2008

Author

Original signed by Briand J. Gaudet

Briand J. Gaudet

Approved by

Original signed by Francine Desharnais for

J. S. Kennedy

Head, Maritime Information and Combat Systems

Approved for release by

Original signed by Calvin Hyatt

Calvin Hyatt

DRP Chair

Abstract

Version 1.3 of the High Level Architecture (HLA) specification was developed by the Defence Modeling and Simulation Office (DMSO) in the 1990s. A more robust standard, IEEE 1516, was proposed in 2000.

Defence R&D Canada – Atlantic's (DRDC Atlantic's) current strategic focus dictates a move towards the newer standard, to allow the Virtual Combat Systems (VCS) group to take advantage of newer tools and an expanded application programming interface (API).

This document describes the differences between HLA 1.3 and IEEE 1516, and the steps required to migrate a federate from 1.3 to 1516. As well, it describes the Polka 2.0 framework, which is a tool used to develop federates that are both version- and vendor-independent.

Résumé

La version 1.3 de la spécification de l'architecture de haut niveau (HLA) a été élaborée dans les années 1990 par le Defence Modeling and Simulation Office (DMSO). Une norme plus robuste, la IEEE 1516, a été proposée en 2000.

L'orientation stratégique actuelle de R et D pour la défense Canada – Atlantique (RDDC Atlantique) exige qu'on adopte progressivement la nouvelle norme afin de permettre au groupe des systèmes de combat virtuel de profiter d'outils plus récents et d'une interface de programmation d'applications (API) améliorée.

Ce document décrit les différences entre la HLA 1.3 et l'IEEE 1516, ainsi que la démarche nécessaire afin de faire migrer les fédérés de la norme 1.3 à la norme 1516. Le document décrit aussi le cadre Polka 2.0, un outil qui sert à développer des fédérés indépendants de la version et des fournisseurs.

Executive Summary

Converting VMSA Federates through Polka 2.0: HLA 1.3 to IEEE 1516.

Gaudet, B.J.; DRDC Atlantic CR 2008-020; Defence R&D Canada – Atlantic; June 2008.

Background: The Virtual Maritime Systems Architecture (VMSA) is a framework for distributed simulations based on the High Level Architecture (HLA) as defined by the Defence Modelling Simulation Office (DMSO). VMSA was originally developed by the Australian Defence Science and Technology Organisation (DSTO) and is now in use by all of The Technical Co-operation Program (TTCP) countries including Canada.

The original HLA specification (HLA 1.3) aimed to describe a federate's interface to the Run Time Infrastructure (RTI), but various vendors have implemented this specification in slightly different ways. The newer IEEE 1516 standard provides more features, flexibility and functionality. Most importantly, it firmly describes an interface to the RTI that must be adhered to by vendors. As a result, a federate that is developed to use the IEEE 1516 standard should be vendor-independent.

Principal results: The differences between HLA 1.3 and IEEE 1516 were explored. A basic methodology for migration from 1.3 to 1516 was designed and implemented in the Polka framework. This required significant changes to the original Polka 1.1 framework and enables all Polka-derived federates to become 1516-compliant. The new version of Polka is referred to as "Polka 2.0".

Significance of results: DRDC Atlantic now has a methodology for the migration of HLA 1.3 federates to the IEEE 1516 standard. As well, the Polka 1.1 framework was modified to allow Polka-derived federates to connect to either a 1.3 or 1516 RTI, independent of the RTI vendor.

Future work: Future work may include extending the Polka 2.0 vendor-specific classes to handle more RTIs, such as PoRTIco and the version-specific classes to handle the upcoming HLA standard referred to as 'HLA Evolved'.

Sommaire

Converting VMSA Federates through Polka 2.0: HLA 1.3 to IEEE 1516.

Gaudet, B.J.; DRDC Atlantic CR 2008-020; R et D pour la défense Canada – Atlantique; juin 2008.

Contexte : L'architecture des systèmes virtuels maritimes (VMSA) est une structure de simulations distribuées qui utilise l'architecture de haut niveau (HLA) telle qu'elle est définie par le Defence Modelling Simulation Office (DMSO). La VMSA a été élaborée en premier lieu par l'Australian Defence Science and Technology Organisation (DSTO) et est maintenant utilisée par tous les pays membres du programme de coopération technique (TTCP), y compris le Canada.

La spécification HLA d'origine (HLA 1.3) avait pour but de décrire l'interface entre les fédérés et l'Infrastructure valorisée à l'exécution (IVA), mais les différents fournisseurs ont adopté cette spécification de façons légèrement différentes. La norme IEEE 1516, plus récente, offre plus de flexibilité et de fonctionnalité. Et surtout, elle décrit de façon formelle l'interface à l'IVA que les fournisseurs doivent adopter. Par conséquent, un fédéré développé selon la norme IEEE 1516 devrait être indépendant du fournisseur.

Principaux résultats : Les différences entre la HLA 1.3 et l'IEEE 1516 ont été étudiées, puis une méthode fondamentale a été conçue et mise en place pour la migration de la norme 1.3 à la norme 1516 à l'aide de la structure Polka. Par conséquent, il a fallu apporter des changements importants à la version originale Polka 1.1, ce qui a permis à tous les fédérés dérivés de Polka de se conformer à l'IEEE 1516. La nouvelle version de Polka se nomme Polka 2.0.

Importance des résultats : RDDC Atlantique possède maintenant une méthode pour la migration de fédérés HLA 1.3 à la norme IEEE 1516. De plus, la structure Polka 1.1 a été modifiée pour permettre aux fédérés dérivés de Polka de se connecter à un IVA 1.3 ou 1516, sans égard au fournisseur d'IVA.

Travaux futurs : Parmi les travaux envisagés dans l'avenir, on pourrait procéder à l'élargissement des classes spécifiques des fournisseurs de Polka 2.0 afin de traiter d'autres IVA, tels que PoRTIco et les classes spécifiques de certaines versions afin de se conformer à la norme prévue « HLA évoluée ».

Table of contents

List of tables

Document revision history

Table 1. *Document Revision History.*

DATE	VERSION	SUMMARY OF CHANGES
6 February 2008	1.0	First release

1. Introduction

To date, there have been two major specifications for the High Level Architecture (HLA): HLA 1.3 and IEEE 1516. Most of the Virtual Maritime Systems Architecture (VMSA) [1]-based federates in use at Defence R&D Canada – Atlantic (DRDC Atlantic) were written to the 1.3 specification. An initiative is underway at DRDC to convert all existing federates to the 1516 standard.

Section 2 of this document examines the differences between the two specifications and describes a methodology for converting from HLA 1.3 to IEEE 1516.

Section 3 of this document provides a technical description of the Polka 2.0 framework. Polka 2.0 is an evolution of the Polka 1.1 framework [2]. Specifically, Polka 2.0 allows any federate developed using Polka to connect to either a 1.3-compliant or a 1516-compliant Run Time Infrastructure (RTI).

The work referred to in the document is essentially independent of the RTI implementation being used. There currently exist several commercial and non-commercial RTI products. Specific support for RTI implementations offered by MÄK Technologies [3] and Pitch [4] has been integrated into Polka 2.0. Other RTI products are easily supported, with minor additional work required to handle any ways in which they may differ from the established 1.3 or 1516 standards.

2. Migration Methodology

The formal specifications for HLA 1.3 and IEEE 1516 can be found in [5] and [6], respectively. The information in this section that refers to specific details of a particular version can be found in these specification documents. However, the 1516 document does not discuss how the 1516 standard differs from the 1.3 standard. Information in this section which is of a comparative nature has been collected and analyzed by the author.

2.1 Differences Between 1.3 and 1516

This section aims to identify any differences between HLA 1.3 and 1516 that a federate developer should be aware of.

2.1.1 Libraries

The 1.3 and 1516 versions of the RTI interface may reside in the same Java library (in the case of MÄK), or in separate ones (in the case of Pitch). In any case, the Java package prefix for 1.3 classes is `hla.rti`, whereas the package prefix for 1516 is `hla.rti1516`. It should be noted that the Defence Modelling and Simulation Office (DMSO) 1.3[1] implementation has a package prefix of `hla.rti13.java1`.

2.1.2 Types

Many of the data items that were represented by integers in 1.3 have been assigned their own dedicated classes in 1516. Table 2 provides a complete list of 1516 data classes and their 1.3 equivalents.

Table 2. *1516 Types and Corresponding 1.3 Types.*

1516 TYPE	1.3 TYPE
AttributeHandle	int
AttributeHandleSet	AttributeHandleSet
AttributeHandleValueMap	ReflectedAttributes
AttributeRegionAssociation	n/a
AttributeSetRegionSetPairList	n/a
DimensionHandle	int
DimensionHandleSet	n/a
FederateAmbassador	FederateAmbassador
FederateHandle	int
FederateHandleRestoreStatusPair	n/a
FederateHandleSaveStatusPair	n/a
FederateHandleSet	FederateHandleSet
InteractionClassHandle	int
LogicalTime	LogicalTime
LogicalTimeInterval	LogicalTimeInterval
MessageRetractionHandle	EventRetractionHandle
MessageRetractionReturn	EventRetractionHandle
MobileFederateServices	MobileFederateServices
ObjectClassHandle	int
ObjectInstanceHandle	int

[1] The DMSO 1.3 standard is less strict than the HLA 1.3 standard.

OrderType	int
ParameterHandle	int
ParameterHandleValueMap	SuppliedParameters
RangeBounds	n/a
RegionHandle	Region
RegionHandleSet	n/a
ResignAction	ResignAction
RestoreFailureReason	n/a
RestoreStatus	n/a
RTIambassador	RTIambassador
SaveFailureReason	n/a
SaveStatus	n/a
ServiceGroup	n/a
TimeQueryReturn	LogicalTime
TransportationType	int

2.1.3 Time Representation

As shown in Table 2, both versions of the RTI specification make use of `LogicalTime` and `LogicalTimeInterval` interface classes for time specification. Even though the class names remained the same, the methods required to implement these interfaces differ significantly, as shown in Table 3.

Table 3. *Comparison of LogicalTime and LogicalTimeInterval Interfaces.*

INTERFACE	METHODS
hla.rti.LogicalTime	void decreaseBy(LogicalTimeInterval i) void increaseBy(LogicalTimeInterval i) LogicalTimeInterval subtract(LogicalTime t) boolean isEqualTo(LogicalTime t) boolean isGreaterThan(LogicalTime t) boolean isGreaterThanOrEqualTo(LogicalTime t) boolean isLessThan(LogicalTime t) boolean isLessThanOrEqualTo(LogicalTime t) void setInitial() void setFinal() void setTo(LogicalTime t) boolean isInitial() boolean isFinal() int encodedLength() void encode(byte[] bytes, int offset)
hla.rti1516.LogicalTime	LogicalTime add(LogicalTimeInterval i) LogicalTime subtract(LogicalTimeInterval i) LogicalTimeInterval distance(LogicalTime t) boolean equals(Object t) int compareTo(Object t) int hashCode() String toString() boolean isInitial() boolean isFinal() int encodedLength() void encode(byte[] bytes, int offset)
hla.rti. **LogicalTimeInterval**	boolean isEqualTo(LogicalTimeInterval i) boolean isGreaterThan(LogicalTimeInterval i) boolean isGreaterThanOrEqualTo(LogicalTimeInterval i) boolean isLessThan(LogicalTimeInterval i) boolean isLessThanOrEqualTo(LogicalTimeInterval i) void setZero() void setEpsilon() void setTo(LogicalTimeInterval t) boolean isZero()

	boolean isEpsilon() int encodedLength() void encode(byte[] bytes, int offset)
hla.rti1516. LogicalTimeInterval	boolean isZero() boolean isEpsilon() LogicalTimeInterval subtract(LogicalTimeInterval i) int compareTo(Object i) Boolean equals(Object i) int hashCode() String toString() int encodedLength() void encode(byte[] bytes, int offset)

2.1.4 Factories

In 1.3, many object classes provided constructors for creating new instances of objects. In 1516, all object instances that are required to call the RTI are created using a factory class within the RTI interface. Each factory class is accessed through a method call in the RTI ambassador. These factory retrieval methods are listed in Table 4.

Table 4. *Factory Retrieval Methods in* `RTIambassador`.

RETRIEVAL METHODS
getAttributeHandleFactory() getAttributeHandleSetFactory() getAttributeHandleValueMapFactory() getDimensionHandleFactory() getDimensionHandleSetFactory() getFederateHandleFactory() getFederateHandleSetFactory() getObjectInstanceHandleFactory() getParameterHandleFactory() getParameterHandleValueMapFactory() getRegionHandleSetFactory()

2.1.5 Miscellaneous

The order of some parameters in RTI ambassador and federate ambassador methods have changed. Fortunately, since these parameters have generally changed from integers to the specific classes used in 1516, errors in parameter order are quickly resolved.

The `tick()` method of the 1.3 RTI ambassador has been replaced by the 1516 `evokeCallback()` and `evokeMultipleCallbacks()` methods.

Many exceptions implemented in HLA 1.3 map directly to identically-named exceptions in IEEE 1516. However, there are quite a few exceptions that do not correspond directly, as shown in Table 5. In addition, there are a set 1.3 exceptions that have no equivalents in 1516 (Table 6), and a set of entirely new 1516 exceptions (Table 7).

Any exceptions deleted from 1.3 are due to a change in interfacing (e.g., `ArrayIndexOutOfBounds` is deleted because 1516 does not use array indices in its method calls).

New 1516 exceptions are a result of new services being offered, or an expansion of functionality from existing 1.3 services.

As well, the DMSO 1.3 Java bindings used `byte` arrays to represent logical time and `Object` for the tag parameter. These have been replaced with `LogicalTime` and `byte` arrays, respectively for both non-DMSO 1.3 (i.e., strict HLA 1.3) and 1516.

Table 5. *Corresponding HLA 1.3 and IEEE 1516 Exceptions.*

HLA 1.3 EXCEPTION	IEEE 1516 EXCEPTION
AttributeNotKnown	AttributeNotRecognized
CouldNotOpenFED	CouldNotOpenFDD
CouldNotRestore	CouldNotInitiateRestore
DimensionNotDefined	RegionDoesNotContainSpecifiedDimension
EnableTimeConstrainedPending	RequestForTimeConstrainedPending
EnableTimeConstrainedWasNotPending	NoRequestToEnableTimeConstrainedWasPending
EnableTimeRegulationPending	RequestForTimeRegulationPending
EnableTimeRegulationWasNotPending	NoRequestToEnableTimeRegulationWasPending
ErrorReadingFED	ErrorReadingFDD
FederateLoggingServiceCalls	FederateServiceInvocationsAreBeingReportedViaMOM
FederationTimeAlreadyPassed	LogicalTimeAlreadyPassed
InteractionClassNotKnown	InteractionClassNotRecognized
InteractionParameterNotKnown	InteractionParameterNotRecognized
InvalidExtents	InvalidDimensionHandle
InvalidFederationTime	IvalidLogicalTime
InvalidOrderingHandle	InvalidOrderType
InvalidRetractionHandle	InvalidMessageRetractionHandle
InvalidTransportationHandle	InvalidTransportationType
ObjectAlreadyRegistered	ObjectInstanceNameInUse
ObjectClassNotKnown	ObjectClassNotRecognized
ObjectNotKnown	ObjectInstanceNotKnown
RegionInUse	RegionInUseForUpdateOrSubscription
RegionNotKnown	RegionNotCreatedByThisFederate
SynchronizationLabelNotAnnounced	SynchronizationPointLabelNotAnnounced
TimeAdvanceAlreadyInProgress	InTimeAdvancingState
TimeAdvanceWasNotInProgress	JoinedFederateIsNotInTimeAdvancingState
TimeConstrainedWasNotEnabled	TimeConstrainedIsNotEnabled
TimeRegulationWasNotEnabled	TimeRegulationIsNotEnabled

Table 6. *Deleted HLA 1.3 Exceptions.*

ArrayIndexOutOfBounds
ConcurrentAccessAttempted
EventNotKnown
FederateNotSubscribed
FederateWasNotAskedToReleaseAttribute
InvalidResignAction
ObjectClassNotSubscribed
SpaceNotDefined

Table 7. *New IEEE 1516 Exceptions.*

AttributeNotSubscribed
AttributeRelevanceAdvisorySwitchIsOff
AttributeRelevanceAdvisorySwitchIsOn
AttributeScopeAdvisorySwitchIsOff
AttributeScopeAdvisorySwitchIsOn
FederateHasNotBegunSave
FederateUnableToUseTime
IllegalName
IllegalTimeArithmetic
InteractionRelevanceAdvisorySwitchIsOff
InteractionRelevanceAdvisorySwitchIsOn
InvalidAttributeHandle
InvalidFederateHandle
InvalidInteractionClassHandle
InvalidObjectClassHandle
InvalidOrderName
InvalidParameterHandle
InvalidRangeBound
InvalidRegion
InvalidServiceGroup
InvalidTransportationName
MessageCanNoLongerBeRetracted
NoAcquisitionPending
ObjectClassRelevanceAdvisorySwitchIsOff
ObjectClassRelevanceAdvisorySwitchIsOn
ObjectInstanceNameNotReserved

2.2 Migration Details

Migration of a Java federate from 1.3 to 1516 was done using the Eclipse development environment, as the Eclipse integrated development environment (IDE) immediately highlights all naming and linkage errors.

2.2.1 Step 1 – Change Libraries

To begin the migration, exclude any 1.3 libraries from the Java federate's project, and replace them with the 1516 versions of the libraries. In the case of the MÄK RTI, both the 1.3 and 1516 classes are contained in the same library. Delete any `import` statements that refer to 1.3 classes or packages. The standard naming of these have the `hla.rti` package prefix. In the case of the DMSO bindings, the 1.3 packages all begin with `hla.rti13.java1`.

At this point, Eclipse will report many errors. In most cases, you can right-click on the indicated error line and choose to `import` the new 1516 class that implements the object you are trying to use. All 1516 packages begin with `hla.rti1516`.

2.2.2 Step 2 –Type Conversion

Convert any references to 1.3 types into the corresponding 1516 types that replace them, as listed in Table 2.

2.2.3 Step 3 – Convert LogicalTime and LogicalTimeInterval

If you are currently using the 1.3 `LogicalTime` and `LogicalTimeInterval` implementation, you will need to modify them to support the 1516 implementations. The different method signatures were shown in Table 3. The classes that implement `LogicalTimeFactory` and `LogicalTimeIntervalFactory` will likewise need to be updated to provide the 1516 versions of both time-related interfaces.

2.2.4 Step 4 – Use the Factory Methods

As previously noted, many of the 1516 types cannot be directly constructed. They are defined as interfaces that are implemented using vendor-specific classes. These types need to be constructed using the factory classes retrieved via methods found in the 1516 version of `RTIambassador` (see Table 4).

2.2.5 Step 5 – Creation of RTI Ambassador

The RTI ambassador used in the federate must also be constructed using a factory. The proper method is:

```
hla.rti1516.RTIambassador ra =
  hla.rti1516.jlc.RtiFactoryFactory.getRtiFactory().getRTIambassador();
```

2.2.6 Step 6 – Replacement of tick()

Any references to the `tick()` method of the 1.3 `RTIambassador` have to be replaced by the `evokeCallback()` or `evokeMultipleCallbacks()` methods of the 1516 version.

2.2.7 Step 7 – Conversion of Exception Classes

Any RTI exceptions raised or handled by your federate will have to be updated to match those used in the 1516 standard. Refer back to Tables 5, 6 and 7 for the appropriate changes.

3. Polka 2.0 Technical Description

3.1 Introduction

Polka 2.0 is a framework that provides a simplified interface to the RTI. Since it acts as a buffer between the federate code and both the RTI ambassador and federate ambassador, Polka 2.0 removes the federate's dependence on both version-specific and vendor-specific code.

Polka 2.0 is an evolution of Polka 1.1, which relied on the DMSO implementation of HLA 1.3. Polka 2.0 was developed using Java (SDK 1.4.2) and the Eclipse 3.1.0 IDE.

3.1.1 Compliance

Table 8 summarizes the compliance for this framework.

Table 8. *Polka 2.0 Compliance.*

	VERSION	COMMENTS
Programming Language	Java (SDK 5.0)	
FOM	Any	
RTI	MÄK 1.3 and 1516, Pitch 1516	
Platform	Windows XP, Intel processors	Since this is a pure Java framework and makes no use of native libraries it should run on any Java compatible platform. It has only been tested on Windows XP.

3.2 Description

Polka 2.0 abstracts the interface that a federate has to the RTI. It automatically reads in the Simulation Object Model (SOM) and uses it to encode and decode data sent to and received from the RTI. Versions 1.3 and 1516 of the RTI interface are abstracted out, so that any Polka 2.0 federate can connect to either version. All vendor-specific code is also integrated into Polka 2.0, so that Polka 2.0 federates are vendor-independent.

All RTI function calls are made by a model layer, which provides a persistent object-oriented view of the federation object model. RTI call-backs (i.e., the federate ambassador functions) are also processed by the model layer.

An additional VMSA layer has been built into Polka 2.0, so that the VMSA synchronizations and tag usage are automated whenever a federate is descended from

the `VMSAFederate` class. Similar domain-specific layers, such as RPR, could also be added. See Figure 1.

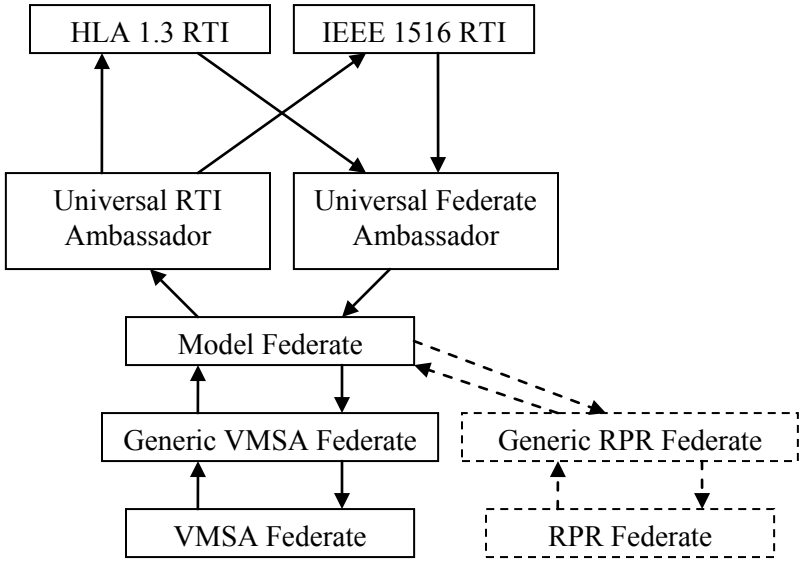

Figure1. *Abstraction Layers in Polka 2.0*

3.3 Functional Description of Models, Classes and Interfaces

The main classes that implement the federate are described below.

3.3.1 UniversalRTIAmbassador

This class implements a common interface to both the 1.3 and 1516 versions of `RTIambassador`. Calls to the methods of this class are redirected to the proper `RTIambassador`, depending on the version specified in the constructor. Any method parameters that changed between the 1.3 and 1516 versions use the universal data classes defined in Polka 2.0 (see `data.Universal*`, below). Method return types are also abstracted using the universal data types.

3.3.2 UniversalFederateAmbassador

This class implements the `FederateAmbassador` interfaces for both 1.3 and 1516 versions of the RTI. As a result, it can act as a federate ambassador for either RTI version. Whenever its call-back methods are called by the RTI, `UniversalFederateAmbassador` repackages the parameters as universal data types (see `data.Universal*`, below) and forwards the call to the `QueuedFederateAmbassador`. To abstract the RTI version being used, 1.3

and 1516 variations of the same call-back call the same `QueuedFederateAmbassador` method.

3.3.3 QueuedFederateAmbassador

This class adds all federate ambassador call-backs into a queue for future processing. This is done to avoid concurrency errors when invoking an RTI ambassador method from within a federate ambassador call-back. The call-back queue is processed after the `tick()` (or `evokeCallbacks()`) method completes. Each call-back method executes a matching `ModelFederateAmbassador` method for processing.

3.3.4 ModelFederateAmbassador

This class moves the incoming federate ambassador call-backs into the data model's context. For example, updates to the attributes of object instances are applied to an internal representation of that object. If the federate has requested to be notified when an object instance is updated, `ModelFederateAmbassador` will handle this.

3.3.5 ModelRTIAmbassador

This class provides the federate with a variety of methods for calling the RTI. The parameters to these methods are generally "model" parameters, that is, they refer to the model's representation of the federation objects, object classes, attributes, parameters and interactions. `ModelRTIAmbassador` translates these representations into universal data types and calls the appropriate `UniversalRTIAmbassador` method.

3.3.6 data.Universal*

These classes represent the various parameters and return types used to call the RTI ambassador and receive call-backs via the federate ambassador. By using the universal types, the federate (and model) can execute in either 1.3 or 1516 mode without knowing which RTI version is in use.

Each universal type can be constructed using either the 1.3 or 1516 version of the type it represents (e.g. either `Integer` or `hla.rti1516.AttributeHandle`, for `UniversalAttributeHandle`). The appropriate data is then accessed by `UniversalRTIAmbassador` via `get13()` or `get1516()` methods, depending on which actual RTI ambassador is being used. Methods in `UniversalFederateAmbassador` also repackage all their parameters as universal types before passing them on to the federate.

3.3.7 UniversalLogicalTime, UniversalLogicalTimeInterval

These two interfaces are used throughout both `UniversalRTIAmbassador` and `UniversalFederateAmbassador`, to represent both 1.3 and 1516 `LogicalTime` and `LogicalTimeInterval` variables. Each interface extends both the corresponding 1.3 and 1516 interfaces. Any class that implements one of these universal interfaces therefore satisfies the requirements of both RTI versions.

3.4 Overall Functional Description

As an application framework, Polka 2.0 performs five significant functions for federates:

- HLA version independence;
- RTI vendor independence;
- Automatic marshalling and de-marshalling of data;
- Queuing of incoming calls to prevent concurrency errors; and
- Encapsulation of federation data in an object model.

3.4.1 HLA Version Independence

Through the use of universal data types, universal time representations and universal RTI interfaces, Polka 2.0 removes any dependency on the HLA version from the federate. The HLA version, either 1.3 or 1516, is passed to the `UniversalRTIAmbassador` on start-up. Thereafter, any version-specific processing is handled by either `UniversalRTIAmbassador` or the version-specific methods of `UniversalFederateAmbassador`.

3.4.2 RTI Vendor Independence

The IEEE 1516 standard is specific enough to guarantee vendor transparency when writing code to interface with a 1516 RTI. There are some variations between vendors in their implementation of the HLA 1.3 specification. The `VendorSpecific` interface and its vendor-specific implementations (`Specific_MaK` and `Specific_Pitch`, as of this writing) allow the development of customized code to deal with these variations. The vendor name is supplied to `UniversalRTIAmbassador` on start-up to choose the appropriate `VendorSpecific` implementation.

3.4.3 Automatic Marshalling and De-marshalling of Data

Neither HLA 1.3 nor IEEE 1516 explicitly define the format of binary data used to send attribute updates and interaction parameters between federates. Polka 2.0 reads in the SOM file to determine the basic data types of both attributes and parameters. The binary transfer format used for these basic data

types is defined in the Federation Agreement. Polka 2.0 allows the developer to include a library of encoder/decoders for these basic data types that conform to the Federation Agreement. Polka 2.0 then automates the encoding and decoding of all federate data, using the basic data types as building blocks.

3.4.4 Queuing of Incoming Calls to Prevent Concurrency Errors

Particularly in HLA 1.3, if a federate makes a call to the RTI during an instance of a `FederateAmbassdor` call-back, a concurrency error is raised. This is a common source of federate errors, particularly amongst inexperienced HLA developers. To prevent this situation, Polka 2.0 inserts all `FederateAmbassador` call-backs into a queue and only processes them after all call-backs have been delivered.

3.4.5 Encapsulation of Federation Data in an Object Model

Polka 2.0 provides an object-oriented view of federation objects and interactions to the federate. This structure is deduced from the SOM file. Any RTI calls from the federate are initiated through this object model, and all federate call-backs are integrated with the model before they are passed on to the federate.

3.5 Integration and Testing

Polka 2.0 federates have been tested in a complex VMSA federation. The Polka 2.0 federates in the federation were Damage[2], Vision[3], JMotion [7], Mogwai [8], JACKBridge [9] and Gunnery [10]. The federation also consisted of the Execution Manager [11] and Horizon [12] federates and was run with the MÄK RTI (version 2.4). The Execution Manager federate was developed by Defence Science and Technology Organisation (DSTO) and works specifically with HLA 1.3 federates, but not IEEE 1516 federates. The Horizon HLA plug-in is also built as an HLA 1.3 federate. Since a VMSA federation cannot be run without the Execution Manager, it is not possible at this time to fully test the success of the 1516 portions of Polka 2.0. However, all of these federates have been executed in an HLA 1.3 federation and all Polka 2.0 federates behaved exactly as required. Since the premise behind adapting Polka federates to work in an HLA 1.3 federation is the same as that behind adapting Polka federates to work in a 1516 federation, it is expected that the main functionality is correctly built. Small, 1516-related bug fixes may however be required once the federation can be fully tested in a 1516 environment.

[2] This federate is new and has not yet been fully documented.
[3] This federate is new and has not yet been fully documented.

3.6 Future Development

As mentioned in Section 3.5, two HLA 1.3-based federates need to be updated to IEEE 1516 before a test of a complete 1516-compliant VMSA federation can be performed. The following are required:

- An IEEE 1516-compliant version of the VMSA Execution Manager federate is required to test any VMSA federation in 1516 mode; and

- An IEEE 1516-compliant version of the Horizon VMSA plug-in is required to run the Horizon federate in a 1516 federation.

Future work may include extending the Polka 2.0 vendor-specific classes to handle more RTIs, such as PoRTIco and the version-specific classes to handle the upcoming HLA standard referred to as 'HLA Evolved'.

4. Progress Towards 1516

With the completion of Polka 2.0, many of the VMSA federates in use at DRDC Atlantic are now IEEE 1516-compliant. In addition to those mentioned in Section 3.5 (i.e., Damage, Vision, JMotion, Mogwai, JACKBridge, and Gunnery), these include:

- Detailer [13], a data logging federate;
- JSound [14], a federate that produces unique sounds for some VMSA interactions;
- COMDAT bridge [15], a federate that allows the Multi-Source Data Fusion (MSDF) component of the Command Decision Aid Technology (COMDAT) project to receive raw tracks and return fused tracks from/to VMSA;
- DRDC ESM [16], a simplistic Electronic Support Measures (ESM) federate based on truth data;
- IFF [17], an Identification Friend Foe (IFF) federate; and
- GCCS [18], a federate which allows tracks to be shared between platforms in Global Command and Control System (GCCS) format.

However, in addition to the VMSA Execution Manager and Horizon VMSA plug-in as previously identified, conversion to 1516 is still outstanding for the following federates which were not developed with Polka:

- SIMDIS [19], a 3-dimensional visual display;
- Tacoma[4], a bridge from VMSA to another federation using a different Federation Object Model (FOM), such as the Real-time Platform Reference (RPR) FOM;
- JBoard [20], a federate developed in-house (based on DSTO's Gameboard federate), which controls the movement of entities along preset paths that are specified by scripts output by the Scenario Generator [21] tool;
- Torpedo[5], a torpedo federate developed in-house for the War-in-a-Box (WIB) exercise;
- Gremlins[23], a generic radar federate developed by DSTO;
- DSTO ESM [24], an ESM federate developed by DSTO;
- Sonar 3 [25], a sonar federate;
- IntAircraft, a federate developed by Defense Technology Agency (DTA) that allows Microsoft Flight Simulator to communicate with VMSA.

[4] Tacoma has undergone various changes since first developed and up-to-date documentation does not currently exist.

[5] Documentation on this federate does not yet exist. However, documentation on the Horizon plug-in which interacts with this federate can be found in [22].

In fact, there are additional federates that are not 1516-compliant which are not listed here since they are outdated and infrequently used. For any of these non-1516 federates, the methodology described in this document can be used to guide the conversion process. While the Execution Manager and Horizon VMSA plug-in will need to be converted in the near future, the remaining federates will most likely be converted on an 'as needed' basis.

Any new federates developed at DRDC Atlantic should use the Polka 2.0 utilities. In this case, all new federates will automatically be both 1.3 and 1516-compliant.

This page intentionally left blank.

References

[1] Canney, S.A. (2002), Virtual Maritime System Architecture Description Document, Issue 2.00, Virtual Maritime System Document Number 00034, Defence Science and Technology Organisation, Edinburgh, Australia.

[2] Dillman, B. (2005), The Polka HLA Utilities, (DRDC Atlantic TM 2005-232), Defence R&D Canada – Atlantic.

[3] MÄK Technologies website: www.mak.com, accessed September 5, 2013.

[4] Pitch website: www.pitch.se, accessed September 5, 2013.

[5] Dynamic Link Compatible HLA API Product Development Group (PDG) (2004), Dynamic Link Compatible HLA API Standard for the HLA Interface Specification Version 1.3, Simulation Interoperability Standards Organization, SISO-STD-004-2004.

[6] Dynamic Link Compatible HLA API Product Development Group (PDG) (2004), Dynamic Link Compatible HLA API Standard for the HLA Interface Specification (IEEE 1516.1 Version), Simulation Interoperability Standards Organization, SISO-STD-004.1-2004.

[7] Hackett, D. and Gaudet, B. (2006), JMotion Federate: User Guide and Technical Description, (DRDC Atlantic CR 2006-026), Defence R&D Canada – Atlantic.

[8] Gaudet, B. (2007), Mogwai Federate: User Guide and Technical Description, (DRDC Atlantic CR 2007-151), Defence R&D Canada – Atlantic.

[9] Gaudet, B. (2008), JACK RedEntities and the JACK-VMSA Bridge Federate, (DRDC Atlantic CR 2008-021), Defence R&D Canada – Atlantic.

[10] Wentzell, T.E. (2006), Gun Control for VBE-E: User Guide and Technical Description, (DRDC Atlantic TM 2006-245), Defence R&D Canada – Atlantic.

[11] Cramp, A. (2000), Virtual Ship Execution Manager, User Guide 1.00, Defence Science and Technology Organisation.

[12] Arnold, A., Goold, L., and Haddy, M. (2005), Horizon 3 Developer Guide, Release 3.3, Published by Innovation Science Pty Ltd., Copyright 2005.

[13] Gaudet, B. (2007), Detailer Federate: User Guide and Technical Description, (DRDC Atlantic CR 2006-239), Defence R&D Canada – Atlantic.

[14] Gaudet, B. (2007), JSound Federate: User Guide and Technical Description, (DRDC Atlantic CR 2006-241), Defence R&D Canada – Atlantic.

[15] Gaudet, B. (2007), COMDAT Federate 2.0: User Guide and Technical Description, (DRDC Atlantic CR 2007-153), Defence R&D Canada – Atlantic.

[16] Gaudet, B. (2007), ESM Federate 1.0 – User Guide and Technical Description, (DRDC Atlantic CR 2007-149), Defence R&D Canada – Atlantic.

[17] Gaudet, B. (2007), IFF Federate 1.0 – User Guide and Technical Description, (DRDC Atlantic CR 2007-148), Defence R&D Canada – Atlantic.

[18] Gaudet, B. (2007), GCCS Federate 1.0 – User Guide and Technical Description, (DRDC Atlantic CR 2007-153), Defence R&D Canada – Atlantic.

[19] Gillis, A. (2005), SIMDIS VMSA Federate: User Guide and Technical Description, (DRDC Atlantic TM 2005-026), Defence R&D Canada – Atlantic.

[20] Gillis, A. (2007), JBoard VMSA Federate: User Guide and Technical Description, (DRDC Atlantic TM 2005-027), Defence R&D Canada – Atlantic.

[21] Roger, W.A. (2003), An Enhanced Scenario Generator for the TTCP Virtual Maritime Systems Architecture, (DRDC Atlantic TM 2003-007), Defence R&D Canada – Atlantic.

[22] Wentzell, T.E. and Goold, L. (2005), Horizon 3 Torpedo Launcher Plug-in: User Guide and Technical Description, (DRDC Atlantic TM 2005-209), Defence R&D Canada – Atlantic.

[23] Munro-Ford, D. (2003), Gremlins Federate User Manual Version 1.2, Defence Science and Technology Organization.

[24] Canney, Shane A. (2001), Constructing an Infrastructure to Facilitate Electronic Support Modelling in the Virtual Ship, DSTO-TR-1159, Defence Science and Technology Organization, Electronic Warfare Division.

[25] Gillis, A. (2007), Sonar 3 VMSA Federate: User Guide and Technical Description, (DRDC Atlantic CR 2005-286), Defence R&D Canada – Atlantic.

List of symbols/abbreviations/acronyms/initialisms

API	Application Programming Interface
COMDAT	Command Decision Aid Technology
DMSO	Defence Modeling and Simulation Office (US Department of Defence)
DRDC	Defence Research and Development Canada
DSTO	Defence Science and Technology Organisation
DTA	Defense Technology Agency
ESM	Electronic Support Measures
FOM	Federation Object Model
GCCS	Global Command and Control System
HLA	High Level Architecture
IFF	Identification Friend Foe
MSDF	Multi-Source Data Fusion
RPR	Real-time Platform Reference
RTI	Run Time Infrastructure
SDK	Software Development Kit
SOM	Simulation Object Model
TTCP	The Technical Co-operation Program
VCS	Virtual Combat Systems
VMSA	Virtual Maritime Systems Architecture
WIB	War-in-a-Box

This page intentionally left blank.

Distribution list

Document No.: DRDC Atlantic CR 2008-020

LIST PART 1: Internal Distribution by Centre:

1 Tania E. Randall
1 Mark G. Hazen
1 Allan Gillis
1 Don Coady
1 Glenn Franck
3 DRDC Atlantic Library

8 TOTAL LIST PART 1

LIST PART 2: External Distribution by DRDKIM

1 DRDKIM
1 Library and Archives Canada

2 TOTAL LIST PART 2

10 TOTAL COPIES REQUIRED

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

SG Software Solutions Incorporated
75 Bristol Avenue, Stillwater Lake, Nova Scotia
B3Z 1E9

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable).

UNCLASSIFIED
(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC APRIL 2011

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title).

Converting VMSA Federates through Polka 2.0: HLA 1.3 to IEEE 1516

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

Gaudet, Briand J.

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information Include Annexes, Appendices, etc).	6b. NO. OF REFS (total cited in document)
June 2008	34	25

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered).

CONTRACT REPORT

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address).

Defence R&D Canada – Atlantic
PO Box 1012
Dartmouth, NS, Canada B2Y 3Z7

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant).	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written).
11bt	W7707-078006/001/HAL

10a ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Atlantic CR 2008-020	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected).

Unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

Version 1.3 of the High Level Architecture (HLA) specification was developed by the Defence Modeling and Simulation Office (DMSO) in the 1990s. A more robust standard, IEEE 1516, was proposed in 2000.

Defence R&D Canada – Atlantic's (DRDC Atlantic's) current strategic focus dictates a move towards the newer standard, to allow the Virtual Combat Systems (VCS) group to take advantage of newer tools and an expanded application programming interface (API).

This document describes the differences between HLA 1.3 and IEEE 1516, and the steps required to migrate a federate from 1.3 to 1516. As well, it describes the Polka 2.0 framework, which is a tool used to develop federates that are both version- and vendor-independent.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

VMSA, distributed simulation, HLA 1.3, IEEE 1516

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE DÉFENSE

www.drdc-rddc.gc.ca