NATO STANDARD

AEP-4754

NATO GENERIC VEHICLE ARCHITECTURE (NGVA) FOR LAND SYSTEMS

VOLUME V: DATA MODEL

Edition A Version 1
FEBRUARY 2018

NORTH ATLANTIC TREATY ORGANIZATION

ALLIED ENGINEERING PUBLICATION

Published by the
NATO STANDARDIZATION OFFICE (NSO)

INTENTIONALLY BLANK

NORTH ATLANTIC TREATY ORGANIZATION (NATO)

NATO STANDARDIZATION OFFICE (NSO)

NATO LETTER OF PROMULGATION

22 February 2018

1. The enclosed Allied Engineering Publication AEP-4754, Volume V, Edition A, Version 1 NATO GENERIC VEHICLE ARCHITECTURE (NGVA) FOR LAND SYSTEMS VOLUME V: DATA MODEL, which has been approved by the nations in the NATO Army Armaments Group, is promulgated herewith. The agreement of nations to use this publication is recorded in STANAG 4754.

2. AEP-4754, Volume V, Edition A, Version 1 is effective upon receipt.

3. No part of this publication may be reproduced, stored in a retrieval system, used commercially, adapted, or transmitted in any form or by any means, electronic, mechanical, photo-copying, recording or otherwise, without the prior permission of the publisher. With the exception of commercial sales, this does not apply to member nations and Partnership for Peace countries, or NATO commands and bodies.

4. This publication shall be handled in accordance with C-M(2002)60.

for [illegible]

Dieter Schmaglowski
Deputy Director NSO
Branch Head P&C

Edvardas MAŽEIKIS
Major General, LTUAF
Director, NATO Standardization Office

INTENTIONALLY BLANK

RESERVED FOR NATIONAL LETTER OF PROMULGATION

INTENTIONALLY BLANK

RECORD OF RESERVATIONS

CHAPTER	RECORD OF RESERVATION BY NATIONS
Note: The reservations listed on this page include only those that were recorded at time of promulgation and may not be complete. Refer to the NATO Standardization Document Database for the complete list of existing reservations.	

INTENTIONALLY BLANK

RECORD OF SPECIFIC RESERVATIONS

[nation]	[detail of reservation]
Note: The reservations listed on this page include only those that were recorded at time of promulgation and may not be complete. Refer to the NATO Standardization Document Database for the complete list of existing reservations.	

INTENTIONALLY BLANK

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1. Purpose

The aim of the NGVA Standard AEP-4754 Volumes I through VII is to enable the member nations to realize the benefits of an open architecture approach to Land vehicle platform design and integration, especially in regard to the vehicle platform electronic data and power infrastructure and the associated safety and verification & validation process.

1.2. Application of the NGVA Standard

The NGVA Standard is to be applied to all future land vehicle platforms and vehicle platform sub-system, as well as current vehicle platform refurbishment and upgrade programmes.

This NGVA Standard is applicable to land vehicle platforms, ranging from simple to complex implementations. The requirements for these implementations are determined by the functionality required of the vehicle platform as a whole system including all sub-systems, and not the automotive or power elements alone. The requirements address equipment to be fitted as part of the initial operating capability and equipment likely to be fitted throughout the life of the vehicle platform. These requirements are expressed in the national system requirements documents and/or the sub-system requirements documents for the individual vehicle platforms concerned.

1.3. Agreement

Ratifying nations agree that the NGVA Standard is to be applied to all future land vehicle platforms and vehicle platform sub-systems, as well as current vehicle platform refurbishment and upgrade programmes. Nations may propose changes at any time to the NATO Standardization Office (NSO).

Germany will act as custodian to maintain Configuration Management (CM) and change management of this Standard and its associated AEP Volumes.

Ratifying nations have agreed that national orders, manuals and instructions implementing this Standard will include a reference to the AEP-4754 Volumes I through VII for purposes of identification.

The NGVA Standard and its associated Volumes I through VII shall be considered as the foundation standard for vehicle sub-system integration, and should any conflict arise between this and other extant NATO documentation, this document shall take precedence.

Deviations from the NGVA Standard shall be agreed by the relevant national procurement office.

1.4. Ratification, implementation, and reservations

Ratification, implementation and reservation details are available on request or through the NATO Standardization Office (NSO) (internet: http://nso.nato.int).

1.5. Feedback

Any comments concerning this publication should be directed to: NATO/NSO – Bvd Leopold III - 1110 Brussels - Belgium.

Proposals for changes and improvements of the NGVA Standard AEP-4754 volumes I through VII shall be sent to the NSO and then forwarded to the custodian who will collect them and will propose new editions of the NGVA Standard AEP-4754 Volumes I through VII.

The NGVA Standard Point-of-Contact as assigned by the NGVA Standard Custodian is BAAINBw K1.2, Ferdinand-Sauerbruch-Str.1, D-56073 Koblenz, Germany.

CHAPTER 2 DEVELOPMENT OF NGVA STANDARD

The NATO Generic Vehicle Architecture (NGVA) Standard was developed under the auspices of the Military Vehicle Association (MILVA).

MILVA is an association of government agencies and industries promoting Vehicle Electronics (Vetronics) in the military environment. MILVA provides an open forum to its members and publishes guidelines and standards on Vetronics issues. MILVA works in close co-operation with NATO through the Land Capability Group on Land Engagement of the NATO Army Armament Group (NAAG).

2.1. NGVA Standard Structure

Figure 1 below illustrates the Standard structure, the Volumes relationships and technical areas covered under each Volume.

NGVA Standard AEP-4754

Volume I:	NGVA Architecture Approach (Describes the NATO Generic Vehicle Architecture (NGVA) concept)
Volume II:	NGVA Power Infrastructure (Defines the design constraints on power interfaces which form the NGVA Power Infrastructure)
Volume III:	NGVA Data Infrastructure (Defines the design constraints on the electronic interfaces that form the NGVA Data Infrastructure)
Volume IV:	NGVA Crew Terminal Software Architecture (Defines the design guidelines and constraints for standardized “Crew Terminal Software Applications”)
Volume V:	NGVA Data Model (Describes the NATO GVA Data Model (NGVA DM), the Model Driven Architecture (MDA) approach used to produce the NGVA DM, the toolset required to produce and manage the configuration control of the NGVA DM and finally the applicability of the NGVA DM to Data Distribution Service (DDS) middleware installed on a GVA compliant platform.)
Volume VI:	NGVA Safety (Outlines the generic procedures to incorporate system safety related planning, development, implementation, commissioning and activities in systems engineering)

Volume VII:	NGVA Verification and Validation (Provides guidance for the verification and validation of NGVA systems regarding their conformity to the AEPs associated with this STANAG)

Figure 1: NGVA Standard AEP-4754

2.2. General Notes

2.2.1. Scope

NGVA is the approach taken by NATO and related industry to standardize the interfaces and protocols for military vehicle systems integration. The Vehicle Architecture (including data and power architectures) is considered as the fundamental enabler that can provide new capabilities on military platforms so as to improve overall effectiveness (including cost) and efficiency within the whole vehicle life cycle. The NGVA Standard does not include standard automotive electronics and power related information.

2.2.2. Warning

National governments, like their contractors, are subject to laws of their respective countries regarding health and safety. Many NATO STANAGs and Standards set out processes and procedures that could be hazardous to health if adequate precautions are not taken. Adherence to those processes and procedures in no way absolves users from complying with their national legal requirements.

2.3. Normative References

The documents and publications shown in Table 1 below are referred to in the text of this AEP Volume. Documents and publications are grouped and listed in alpha-numeric order:

1. DDS Interoperability Wire Protocol Specification v2.1	DDS Interoperability Wire Protocol specification (DDS-RTPS) (http://www.omg.org/cgi-bin/doc?formal/10-11-01.pdf)
2. IDL version 3.5	Interface Definition Language http://www.omg.org/spec/IDL35/3.5
3. MDA Guide revision 2.0	Model Driven Architecture http://www.omg.org/cgi-bin/doc?ormsc/14-06-01
4. OMG Data Distribution Service (DDS) v1.2	Data Distribution Service for Real-Time Systems ('DDS) (http://www.omg.org/cgi-bin/doc?formal/07-01-01)
5. STANAG 4697/AEP-79	Platform Level Extended Video Standard (PLEVID)
6. UML Version 2.0 Infrastructure Specification	http://www.omg.org/spec/UML/2.0/Infrastructure/PDF
7. UML Version 2.0 Superstructure	http://www.omg.org/spec/UML/2.0/Superstructure/PDF

Specification	

Table 1: Normative References

Reference in Standard AEP-4754 and its Volumes to any normative references refers to, in any Invitation to Tender (ITT) or contract, the edition and all amendments current at the date of such tender or contract, unless a specific edition is indicated. For some standards, the most recent editions shall always apply due to safety and regulatory requirements.

In consideration of the above and as best practice, those setting the requirements shall be fully aware of the issue, amendment status and application of all normative references, particularly when forming part of an ITT or contract.

2.4. Conventions

For the purposes of all AEP Volumes all requirements are specifically detailed in tables with each requirement classified as in the paragraph 2.6. Where an AEP Volume contains no specific requirement tables they should serve as implementation guidance until technical standardization requirements are developed and included.

2.5. Requirements Classifications

The following classifications are to be used for all NGVA related requirements.

2.5.1. Compulsory Requirement (CR)

The requirement needs to be implemented in order to conform to Standard AEP-4754 and to gain certification. Compulsory requirements are listed in the Requirements Tables inside the AEPs and marked as “CR”.

2.5.2. Optional Enhancement (OE)

Optional Enhancements do not need to be implemented in order to conform to Standard AEP-4754. However, if such a capability is present, it needs to be implemented according to the stated specification in order to be compliant. Optional Enhancements are listed in the Requirements Tables inside the AEPs and marked as “OE”.

2.6. Abbreviations

Abbreviations referred to in this AEP Volume are given in Annex A.

2.7. Terms and Definitions

2.7.1. NGVA Definitions

1. **Base Vehicle**: The basic vehicle structure and those systems needed to enable it to perform its automotive functions and mobility. Where fitted it also includes those systems needed to control turrets and other physical elements e.g. a mine plough.
2. **Base Vehicle Sub-System**: A system that forms part of the base vehicle
3. **Electronic Architecture**: The combination of the electronic based sub-systems and electronic infrastructure that supports the vehicle crew to undertake their operational tasks

4. **NATO Generic Vehicle Architecture (NGVA):** The term ‘NATO Generic Vehicle Architecture’ refers to the open, modular and scalable architectural approach applied to the design of vehicle platforms.
5. **Hard Switching:** The ability to control or operate a sub-system using physically based means.
6. **Measure of Effectiveness:** A description of how effective a solution candidate is for a particular assessment criterion.
7. **Measure of Performance:** A statement that describes the assessment criterion or criteria needed to satisfy a given requirement.
8. **Modular**: A modular architecture is designed in such a way as to allow the replacement or addition of sub-systems and upgrades as required without any undesirable emerging properties.
9. **NGVA Compliant:** NGVA Compliance applies to the whole vehicle platform and means that any sub-system existing on the platform complies with the requirements defined in STANAG 4754 and associated AEPs.
10. **NGVA Electronic Infrastructure:** The physical cables and connectors that provide means of distributing data around a base vehicle. It also includes any enabling logical components and functions e.g. Core platform management software, interface software, transport protocols and message definitions.
11. **NGVA Power Infrastructure:** The physical cables, connectors and other components that provide the means of distributing and controlling electrical power around a vehicle platform.
12. **NGVA Ready**: NGVA Ready applies at a sub-system level and means that sub-systems and components have been developed to a level where they can be efficiently integrated within a “NGVA Compliant” whole vehicle Electronics. This would mean passing an incremental process with two sequentially-related Compatibility levels:
    a. **Connectivity Compatibility**: Ensures that the (sub-) system can be physically integrated into the NGVA architecture without any negative impacts to existing NGVA components. Physical power and network interfaces comply with the requirements of Power and Data Infrastructure AEPs.
    b. **Communication Compatibility**: Connectivity Readiness and data interfaces (DDS/Video) with associated NGVA Data Model implementation that comply with the requirements of Data Model and Data Infrastructure AEPs.
13. **Operator:** Any person required to interface and control vehicle platform sub-systems.
14. **Power Management:** The means of prioritizing and controlling the electrical power loads throughout the vehicle platform.
15. **Scalable**: The trait of a system in being able to scale in order to handle increased loads of work.
16. **Soft Switching:** The ability to control or operate a sub-system using software functionality.
17. **Sub-System:** Separable elements or collections of equipment or software added to a base vehicle that provides operationally required capabilities over and above those delivered by the base vehicle.

18. **System:** A combination, with defined boundaries, of elements that are used together in a defined operating environment to perform a given task or achieve a specific purpose. The elements may include personnel, procedures, materials, tools, products, facilities, services and/or data as appropriate.
19. **Vehicle Crew:** All personnel located in the vehicle platform with defined roles needed to fulfil the necessary operational functions.
20. **Vehicle Platform**: The vehicle and all its integrated sub-systems.
21. **Vehicle Users:** The individuals and groups of people who interact locally to operate, support, sustain, maintain or otherwise interface directly with the Vehicle Platform and its sub-systems. It includes Service personnel, Reserve personnel, and Civilian employees, and may include personnel under other service supply contracts.

2.7.2. AEP Specific Definitions

1. **Class**: A Class is an element of a Class Diagram; one of the diagram types that form UML.
2. **CI:** A Configuration Item (CI) is a component of a system that is treated as a self-contained unit for the purposes of identification and change control. All CIs are uniquely identified by CI version numbers. A CI may be a primitive system building block (e.g. code module) or an aggregate of other CIs (e.g. a sub-system is an aggregate of software units).
3. **DDS:** Data Distribution Service is a type of middleware that uses the Publish/Subscribe paradigm. It is governed by the Object Management Group (OMG).
4. **IDL:** Interface definition language is a specification language used to describe a software component's interface.
5. **Middleware**: Software that acts to abstract application software from the hardware/software infrastructure.
6. **Model Driven Architecture**: MDA is an open specification for software generation management by the Object Management Group (OMG). For more details on MDA see the OMG FAQs (http://www.omg.org/mda/faq_mda.htm).
7. **NGVA Data Model**: A NATO specific release of the Data Model formed from modules contained within the Subversion repository.
8. **PIM**: A Platform Independent Model is a UML model which is independent of any software platform.
9. **PSM**: A Platform Specific Model is a UML model which includes information relating to a specific software platform.
10. **QoS:** DDS topics are assigned a Quality of Service on a per topic basis. The Quality of Service governs the way in which that topic is handled by a DDS system thereby allowing tuning of the overall system.
11. **Topic:** A topic is a DDS data structure that has a name, a number of attributes and has an associated Quality of Service.
12. **UML**: Unified Modeling Language is an open specification for software modeling issued by the Object Management Group (OMG).

INTENTIONALLY BLANK

CHAPTER 3 NATO GENERIC VEHICLE ARCHITECTURE DATA MODEL

3.1. NGVA Data Model Introduction

The NGVA Data Model is the expression of the system information needs for a NATO land vehicle, stated in a technology independent way, and provides the means to automatically generate technology specific data interfaces for vehicle subsystems. The data interfaces created can then be added to subsystem software applications embedded on a vehicle platform that supports standardized data distribution over an Ethernet network. The NGVA Data Model is a set of jointly developed agreed upon modules that have achieved the desired level of maturity to be part of a given Version of the Standard.

The NGVA Data Model defines the data structure and format to be used by sub-systems and components communicating via Data Distribution Service (DDS) middleware installed on a compliant land platform.

The components on each NGVA compliant platform will implement all the modules or a subset of the Data Model modules as appropriate to its requirements.

The current Standard or agreed upon Version of the Data Model and the NGVA Translator will always be hosted at the NGVA Home Page. The United Kingdom, however will host the single Data Model repository from which all future NATO Data Model releases will be built and controlled. Model Driven Architecture (MDA)

3.1.1. Introduction

Definition of the data communication need is achieved by a modeling approach using the Unified Modeling Language (UML). UML models lead to highly structured and repeatable data Interface Definition Language (IDL). These definitions can then be compiled into executable code. The chain that leads from a high level UML model to interface definitions capable of being used to generate software code is called a Model Driven Architecture (MDA). If all vehicle electronic architectures use the same data definitions then a level of data compatibility between components will be achieved.

3.1.2. Brief description of MDA

MDA is an open specification managed by the Object Management Group (OMG). It is a way of developing applications and writing specifications, based on a computing Platform Independent Model (PIM) of the application or specification's business functionality and behavior. A complete MDA specification consists of a definitive computing platform-independent base model, plus one or more computing Platform Specific Models (PSM) and sets of interface definitions, each describing how the base model is implemented on a different middleware platform. A complete MDA application consists of a definitive PIM, plus one or more PSMs and complete implementations, one on each computing platform that the application developer decides to support.

3.1.3. Use of MDA in NGVA

A MDA approach to development has been adopted for the NGVA Data Model. Using this approach the data that needs to move between vehicle sub-systems is initially modeled in a computing platform independent way i.e. abstracted away from any underlying technology implementation. Figure 1 shows the elements of the NGVA Data Model MDA tool chain.

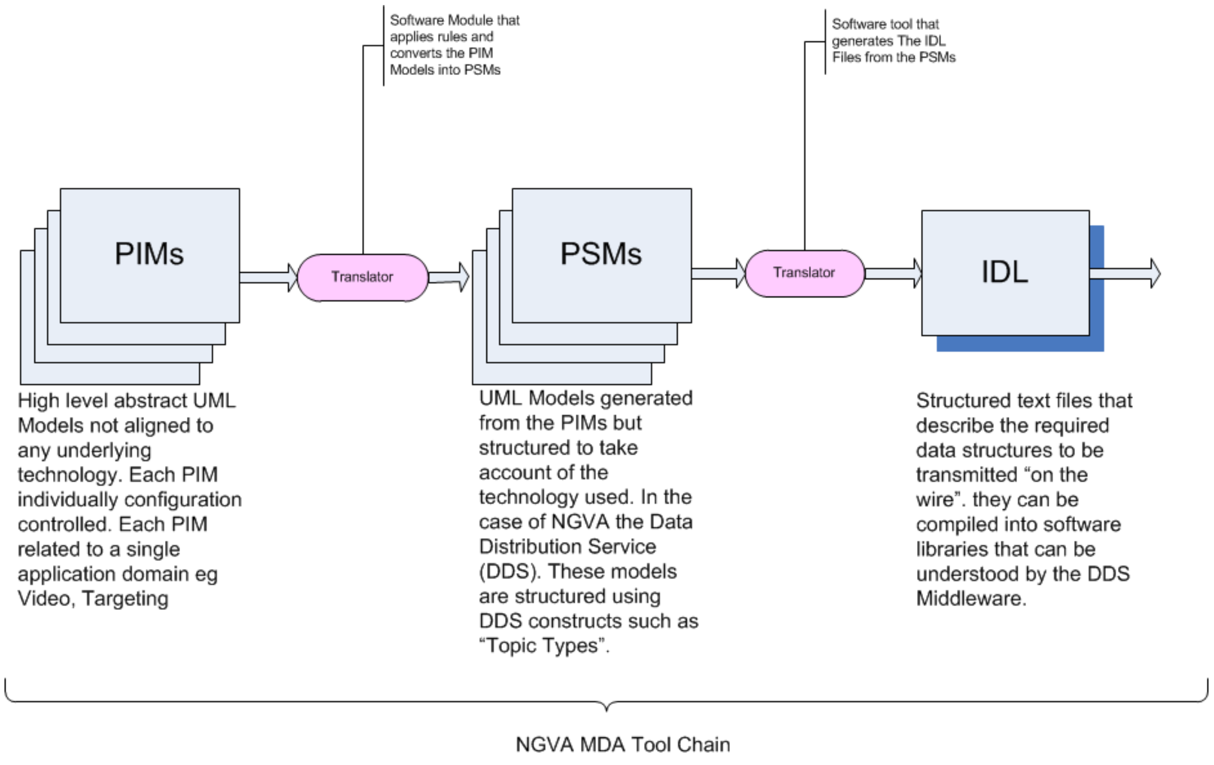

Software Module that applies rules and converts the PIM Models into PSMs
Software tool that generates The IDL Files from the PSMs
PIMs
Translator
PSMs
Translator
IDL
High level abstract UML Models not aligned to any underlying technology. Each PIM individually configuration controlled. Each PIM related to a single application domain eg Video, Targeting
UML Models generated from the PIMs but structured to take account of the technology used. In the case of NGVA the Data Distribution Service (DDS). These models are structured using DDS constructs such as "Topic Types".
Structured text files that describe the required data structures to be transmitted "on the wire". they can be compiled into software libraries that can be understood by the DDS Middleware.
NGVA MDA Tool Chain

Figure 1: MDA tool chain

3.1.4. Interface Definition Language (IDL)

The OMG Interface Definition Language (IDL) is the language used to describe the data interfaces. An interface definition written in IDL completely defines the interface and fully specifies each operation's parameters. An IDL interface provides the information needed to develop software applications that use the interface's operations.

The client software interfaces are not written in IDL, which is purely a descriptive language, but in languages for which mappings from IDL concepts have been defined. The mapping of an IDL concept to a software language will depend on the facilities available in the software language. The IDL files can be converted and used with a number of different software languages such as Java and C++.

CHAPTER 4 DATA MODEL TOOLSET

In order to manage the development, maintenance and configuration control of the Data Model, a number of software tools are employed. These software tools fall into two categories:

1. Tools for Data Model development and maintenance
2. Tools for Data Model Configuration Management

4.1. Tools for Data Model development and maintenance

4.1.1. Rational Rhapsody

The Data Model has been created using IBM Rational Rhapsody Architect for Systems Engineers version 8.1.1.

Rational Rhapsody may be used to open a copy of the Data Model provided it is version 8.1.1 or later. Changes to the model will be made in version 8.1.1 unless otherwise stated.

4.1.2. Enterprise Architect

Enterprise Architect (EA) may be used to view a copy of the Data Model provided that it is version 9.2 or later. In order to view a copy of the model using Enterprise Architect (EA), the Rhapsody .rpy file must be imported into Enterprise Architect.

Note: The XML Metadata Interchange (XMI) interface which should provide the ability to exchange models between these tools does not function correctly. Therefore, while it is possible to view the model in EA (using the import function), it is not possible make changes in EA that can be exported back to Rational Rhapsody.

4.1.3. NGVA Translator

Section 0 describes the MDA approach to developing the Data Model. The latest NGVA Data Model Translator will always be available at the NGVA Web Site. It is fundamental to this approach that PIM modules developed for the Data Model and placed under configuration control are the Configuration Items (CIs) that are modified when changes are sanctioned by the Change Management process. Hence, artifacts such as PSMs and IDL must be generated automatically from the PIM.

4.2. Tools for Data Model Configuration Management

4.2.1. Subversion

Subversion is an Open Source Version Control System (VCS). Subversion manages files and directories, and the changes made to them, over time, thereby allowing recovery of older versions of data or examination of the history of how that data changed.

Subversion can operate across networks, allowing it to be accessed by users at different locations. Providing the ability for users to modify and manage the same set of data from their respective locations fosters collaboration. Progress can occur more quickly without a single conduit through which all modifications must occur. Since all work is versioned, an incorrect change made to the data, can be easily reversed.

Subversion is not a Software Configuration Management (SCM) system; it is a general system that can be used to manage any collection of files.

Subversion is Open Source software provided by Apache Software Foundation. The hosted Data Model website is currently running Subversion version 1.7.

4.2.2. Trac - project management tracking system

Trac is Open Source software provided by Edgewall Software. The hosted Data Model website is currently running Trac version 1.0.

Trac is an issue tracking system for software development projects. Trac provides an issue tracking system based on tickets. It uses a minimalistic approach to web-based software project management and aims to impose a marginal overhead on established development process and policies.

Trac provides an interface to Subversion (section 4.2.1), an integrated wiki and convenient reporting facilities. Trac provides wiki mark-up in issue descriptions and commit messages, creating links and seamless references between bugs, tasks, change sets, files and wiki pages.

CHAPTER 5 DATA MODEL CONFIGURATION & ACCESS

5.1. Data Model Configuration Items

The Data Model consists of a number of modules where each module is detailed in a PIM that relates to an area of concern or modelling domain. The module is a Configuration Item (CI) and is the lowest level at which configuration control is applied.

The Data Model is also a configuration item and configuration control is applied at this level in addition to that at the module level.

5.2. Data Model Configuration

Consider a Data Model that consists of modules A, B, C, D & E. Since each module is a CI, it will be assigned a specific version number. The complete Data Model, also a CI, will be assigned a specific version number that has an associated Release Note. The Release Note will specify the Data Model version and the modules, along with their version numbers, that form the Data Model release.

A Data Model 1 release consisting of modules A, B, C & D is illustrated in the Figure 2 below.

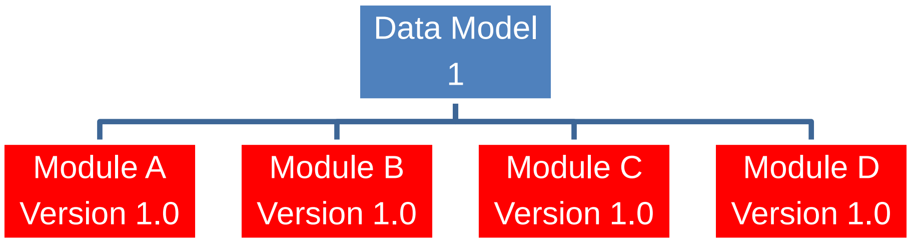


Figure 2: Data Model Version 1.0

Compare this with a Data Model 2 release consisting of modules A, B, C & E shown in Figure 3.

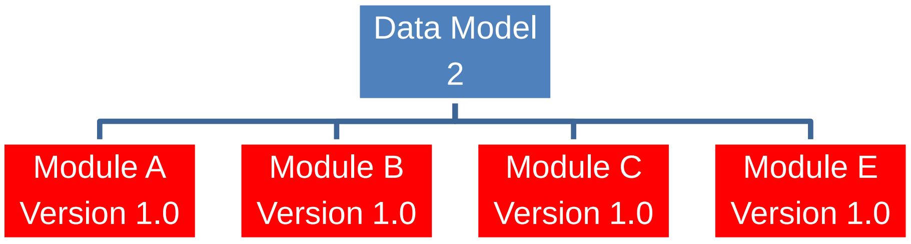


Figure 3: Data Model Version 2.0

The two Data Model releases will be given different version numbers because they are created from a different set of modules.
Now consider a further Data Model 3 release that consists of modules A, B, C & E but with module A at version 1.1, shown in Figure 4. This data model version will also be given a different version number since module A is at version 1.1.

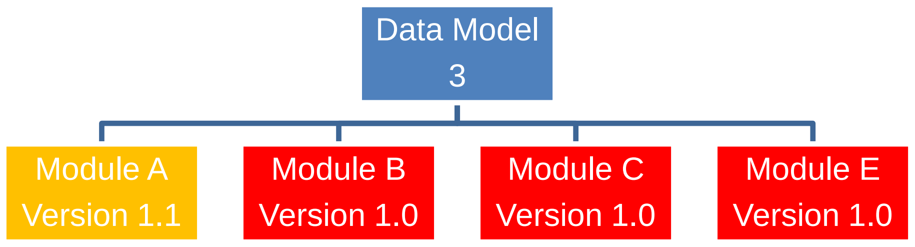


Figure 4: Data Model Version 3.0

In summary, Data Models are released with a version number and a Release Note. The Release Note for that version will detail the modules (and their version numbers) that constitute that version.

5.3. Data Model Repository

The data model is held in a subversion repository and is available to authorized users on a read only basis. The structure of the subversion repository is explained below.

The Repository top level[1] and immediate folders are shown in Figure 9.
At this level, the repository contains 4 folders, highlighted in red in Figure 9. The function of each of these folders is as follows:

1. build_sets – a folder that essentially contains released versions of Data Models i.e NGVA, the UK GVA, the UK GSA[2] or the UK GBA[3].
2. common_modules – a folder containing Data Model modules that may be widely applied. Modules in this category usually contain data types that are used by many modules; hence they are defined in a single module and referenced by other modules.
3. domain_modules – a folder containing Data Model modules that represent a single modelling domain (see 3.2.4). Modules in this category may be used as building blocks to construct a complete Data Model which is then released into the 'build_sets' folder.

[1] The data model is currently held in a subversion repository – https://repository.landopensystems.mod.uk/data/svndata/datamodel
Subversion is an Open Source version control system that manages files and directories, and the changes made to them, over time, thereby allowing recovery of older versions of data or examination of the history of how that data changed.
[2] GSA – The UK Generic Soldier Architecture, Def Stan 23-12
[3] GBA – The UK Generic Base Architecture, Def Stan 23-13

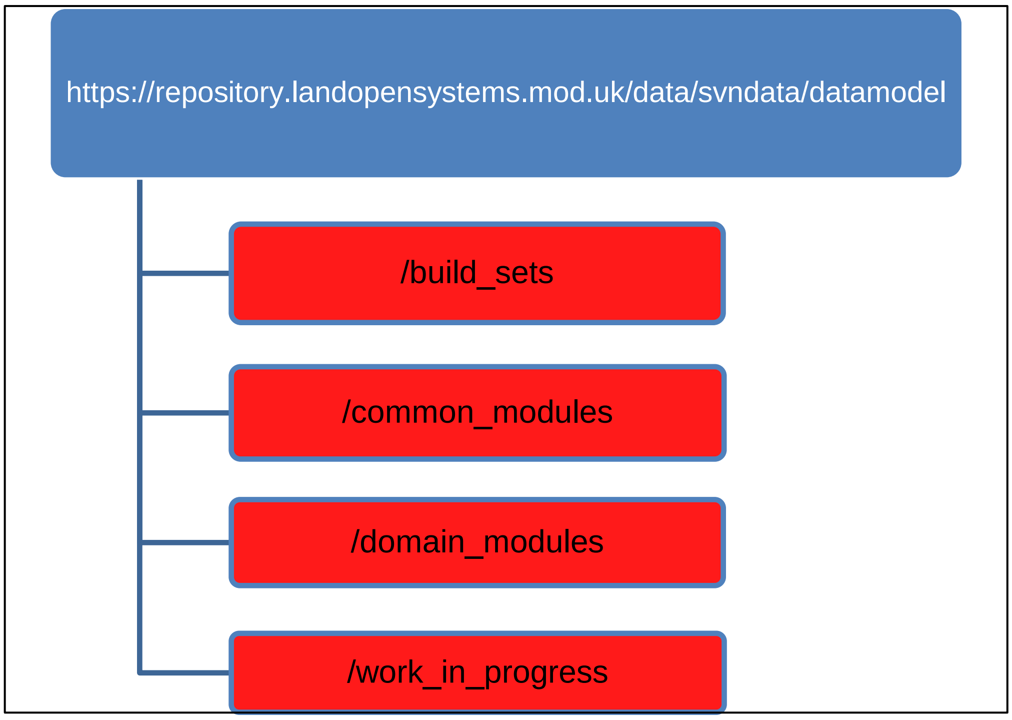

https://repository.landopensystems.mod.uk/data/svndata/datamodel
/build_sets
/common_modules
/domain_modules
/work_in_progress

Figure 5: Repository levels 1& 2

4. work_in_progress – this folder is essentially a ‘scratch pad’ area for module development. Modules held in this folder are not under formal configuration control and are therefore subject to frequent and undocumented changes. Once a module development has been completed, it will be moved to the domain_modules folder and subjected to formal configuration control.

The build_sets folder of the repository will contain a folder for each ‘Architecture’ held, such that different Data Model builds are possible using the same fundamental building blocks or modules. This is illustrated in Figure 6 where released versions of Data Models are grouped according to the Architecture to which they apply.

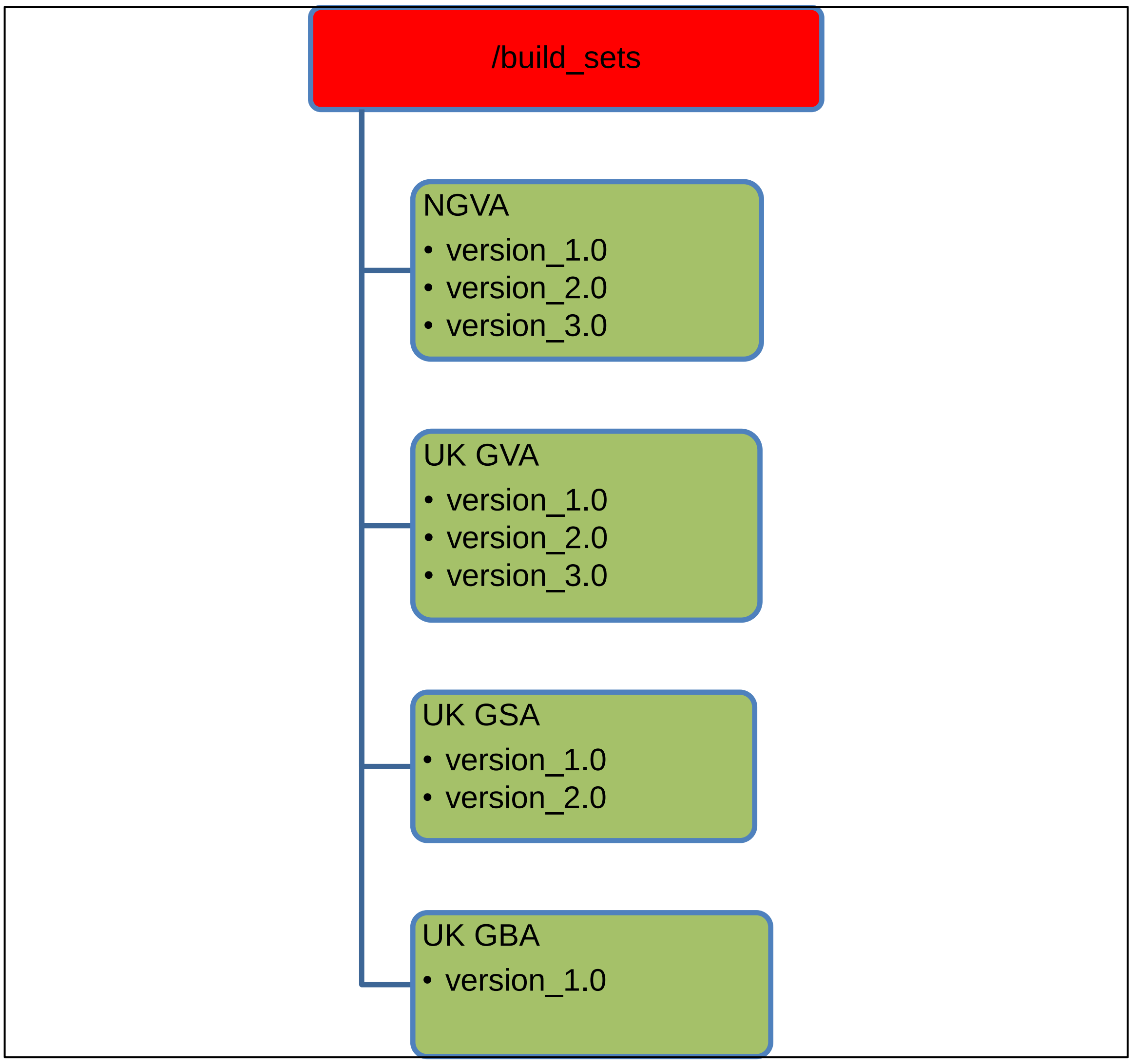


Figure 6: Repository build sets level

The common_modules folder of the repository (see Figure 7) contains a folder for each common module held. Since a module is a Configuration Item, another level of folders exists below each module in order to facilitate the configuation control, these are:

1. branches – contains a version of a module that may contain differences from the version held in the trunk. Once a branch is regarded as complete it is usually merged into the trunk and a new release of that module is created and placed in the tags folder.
2. tags – contains all released version of the parent module.
3. trunk – contains the latest version of the module. This should generally be identical to the latest released version of the module held in the tags folder (this can be confirmed by comparing the revision number for the trunk against the revision number for the latest released version in tags).

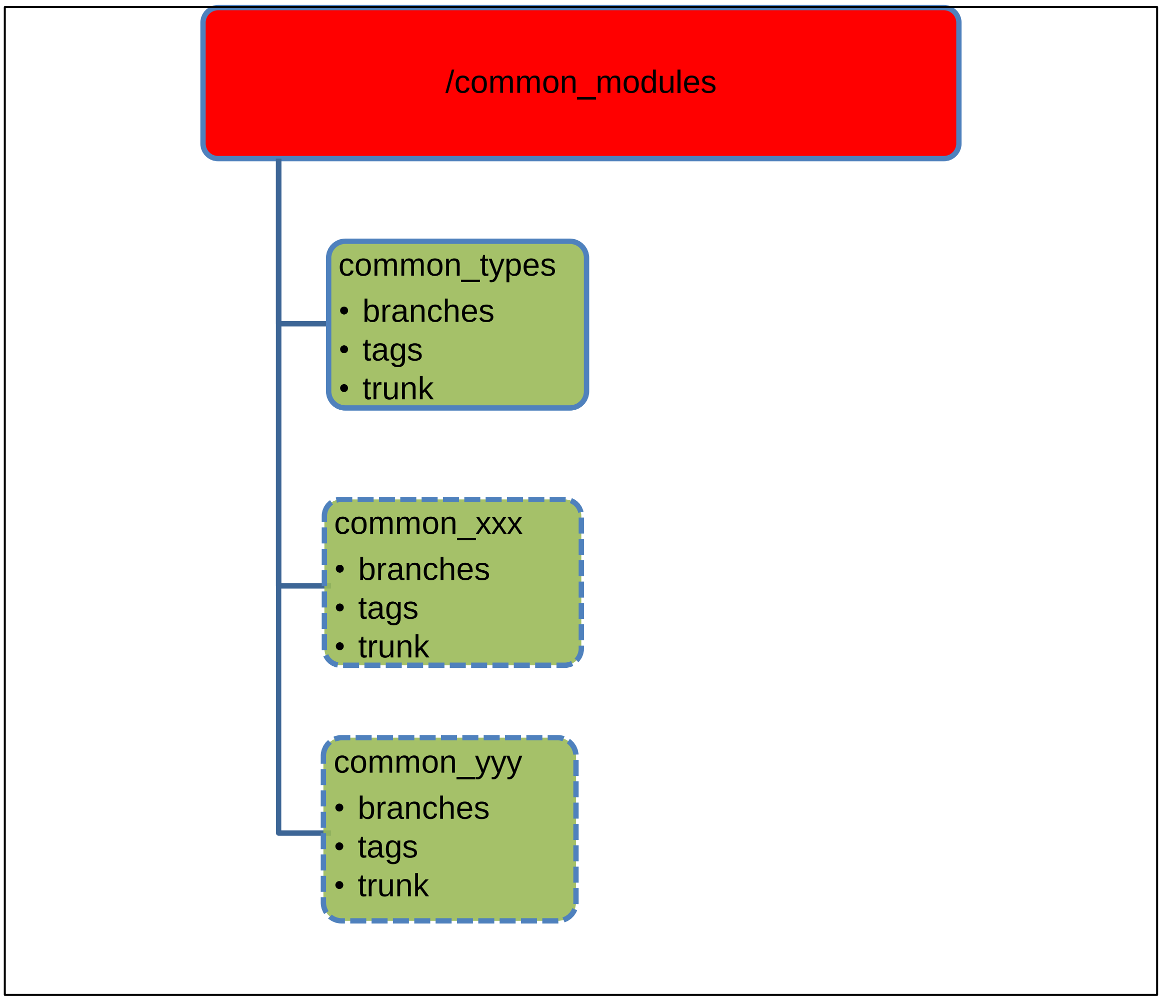


Figure 7: Repository common_modules level

The domain_modules folder of the repository (see Figure 8) contains a folder for each domain module held. Domain modules are also Configuration Items, hence a further level of folders exists below each module in order to facilitate the configuation control, these are:

1. branches – contains a version of a module that may contain differences from the version held in the trunk. Once a branch is regarded as complete it is usually merged into the trunk and a new release of that module is created and placed in the tags folder.
2. tags – contains all released version of the parent module.
3. trunk – contains the latest version of the module. This should generally be identical to the latest released version of the module held in the tags folder (this can be confirmed by comparing the revision number for the trunk against the revision number for the latest released version in tags).

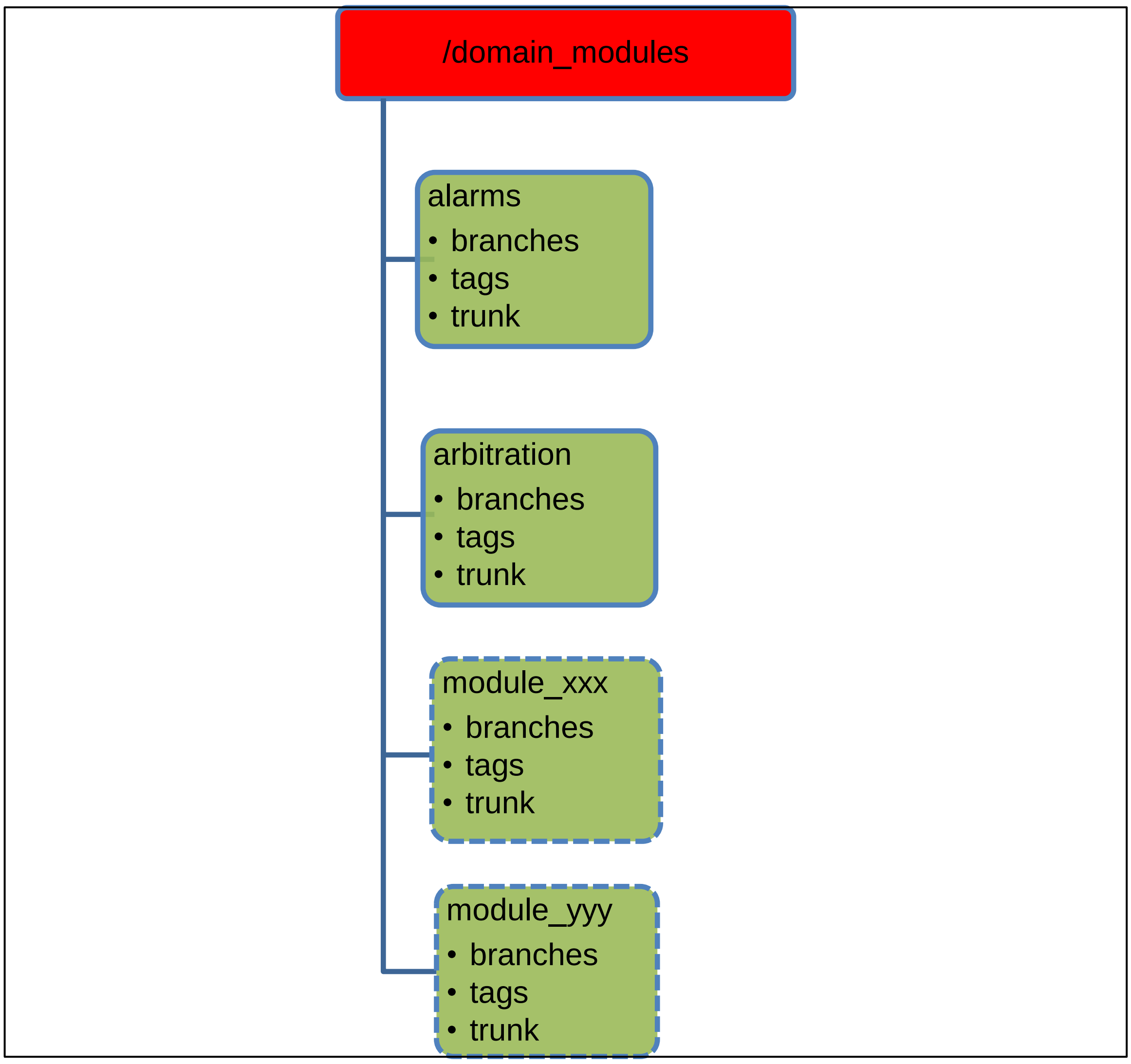


Figure 8: Repository domain_modules level

5.4. Data Model Access and Further Information

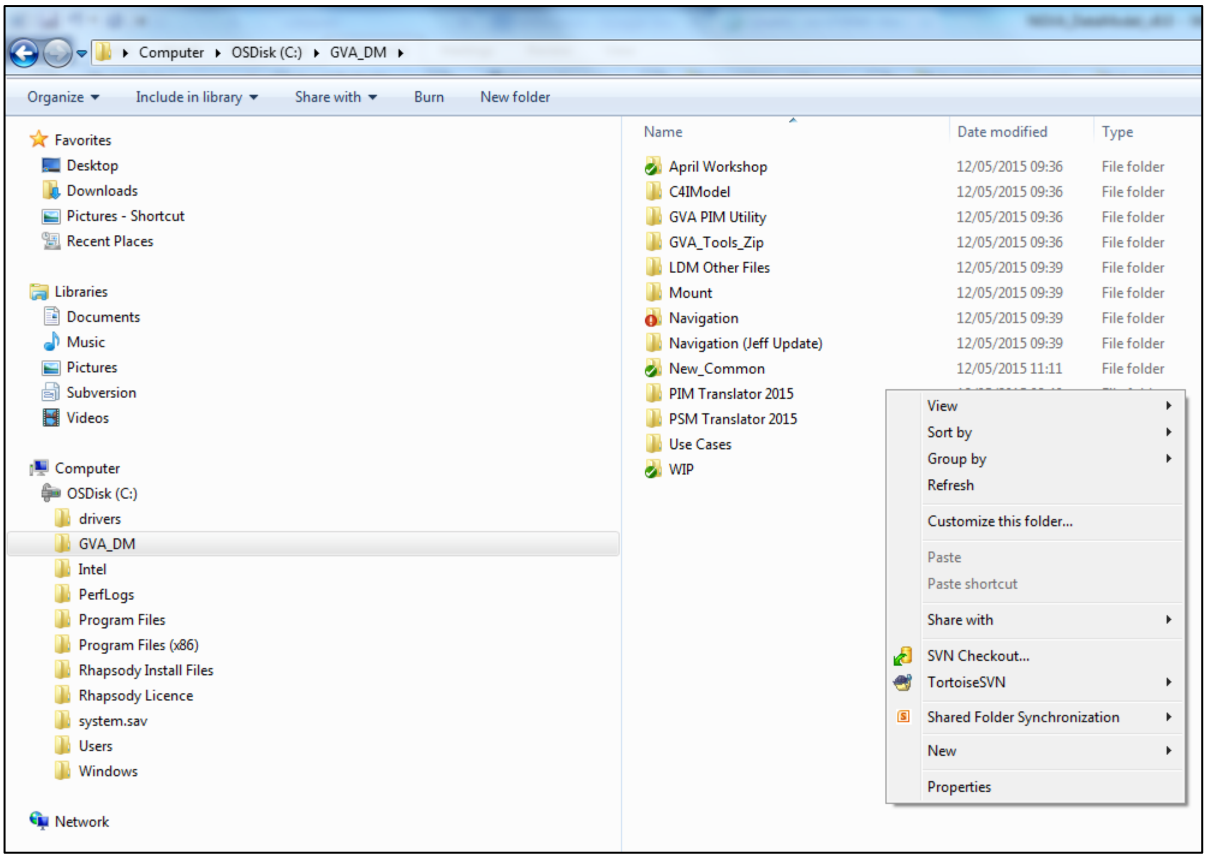
The current version of the NGVA Data Model and the NGVA Data Model translator are accessible from the internet at http://www.natogva.org. Explanatory information regarding the modeling approach (Land Data Model Methodology and Module Development (NGVA Model Maturity Process Guidelines and Checklists)) can also be found at this address.

Access to the full Data Model repository containing NGVA and non NGVA Modules are accessible from the Internet at https://repository.landopensystems.mod.uk/data/svndata/datamodel. Both Internet Sites require registration.

5.4.1. Access via a Subversion Client

In order to gain access, the user must have installed a subversion client such as TortoiseSVN [4]on a Computer that can access the Internet.

Once installed, TortoiseSVN is activated by right clicking in an Explorer window. This action brings up the context menu from which 'TortoiseSVN' and then 'Repo-browser' should be selected. In the dialog box that is displayed, the URL repository https://repository.landopensystems.mod.uk/data/svndata/datamodel should be entered. Figure 9 below shows the invocation of TortoiseSVN.



Name	Date modified	Type
April Workshop	12/05/2015 09:36	File folder
C4IModel	12/05/2015 09:36	File folder
GVA PIM Utility	12/05/2015 09:36	File folder
GVA_Tools_Zip	12/05/2015 09:36	File folder
LDM Other Files	12/05/2015 09:39	File folder
Mount	12/05/2015 09:39	File folder
Navigation	12/05/2015 09:39	File folder
Navigation (Jeff Update)	12/05/2015 09:39	File folder
New_Common	12/05/2015 11:11	File folder
PIM Translator 2015		
PSM Translator 2015		
Use Cases		
WIP		

Figure 9: Invocation of TortoiseSVN

5.4.2. Access via a web browser

It is possible to access the repository via a browser such as Chrome or Firefox by entering the URLs in section 5.4.1 directly. This method is not recommended since it is cumbersome and liable to result in errors.

[4] The version of Subversion running on the Host should match the version of TortoiseSVN installed on the Client, hence version 1.7.x should be installed on the Client.

INTENTIONALLY BLANK

CHAPTER 6 NATO GVA DATA MODEL CHANGE CONTROL

6.1. Data Model Custodian

The United Kingdom is responsible for maintenance of the NGVA Data Model and for making changes to the Data Model when and as instructed by the NATO Change Control Board (CCB).

6.2. Data Model Version

The NGVA Reference Version 1.0 of the Data Model is detailed on the NGVA Home Page http://www.natogva.org. This version contains the set of modules from which an NGVA compliant platform can be constructed. Future NGVA approved versions will also appear on the NGVA home page.

6.3. Data Model Change Control Board

The Change Control Board (CCB) for approval/rejection of changes to the NGVA Data Model shall consist of a Chairman and representation from NATO nations. This group will be convened as determined by the appointed CCB Chairman.

6.3.1. Data Model CCB Chairman

The Chairman shall be appointed by the NGVA Standard Custodian and confirmed by the NGVA Standard Management Group.

6.3.2. CCB Quorum

CCB shall be regarded as quorate provided that 4 or more representatives from NATO nations are present, not including the group Chairman.

6.4. Configuration Control

The configuration control system is required to maintain traceability of changes made to the NGVA Data Model. The CCB will ensure that all requested changes do not impact on the Module Maturity Level (MML) by lowering the module's MML below MML 3. Since even minor changes to the module (such as spelling errors) can result in the generation of different IDLs, the CCB will be responsible for all changes to NGVA Data Model modules. The major input to the configuration control system is the Trac ticket system. A new ticket may be raised by anyone that has authorized access to the NGVA Data Model Website. The change requestor shall suggest a priority based on his needs.

There are three reasons to raise a new ticket, namely:

1. A defect has been discovered,
2. An enhancement to the Data Model or one of its modules has been identified,
3. A task to be carried out on the Data Model or one of its modules has been identified.

Once the ticket has been raised it shall be recorded within the ticket database and be visible via the various reports available on the Land Data Model website[5]. The

[5] https://gvawiki.landopensystems.mod.uk/trac/report

ticket will then effectively enter the ticket workflow state diagram (see Figure 10) at the ‘new’ state.

The CCB Chairman is the responsible Point of Contact for all NGVA Data Model tickets. The CCB Chairman will inform the CCB of ticket priorities and status.

Once the ticket has been assigned an owner, discussions and solutions can be achieved via ticket updates until a convergent solution is reached. At this point the ticket shall be considered by the Change Control Board. Table 1 describes the ticket workflow state/transition descriptions.

6.5. Ticket Workflow

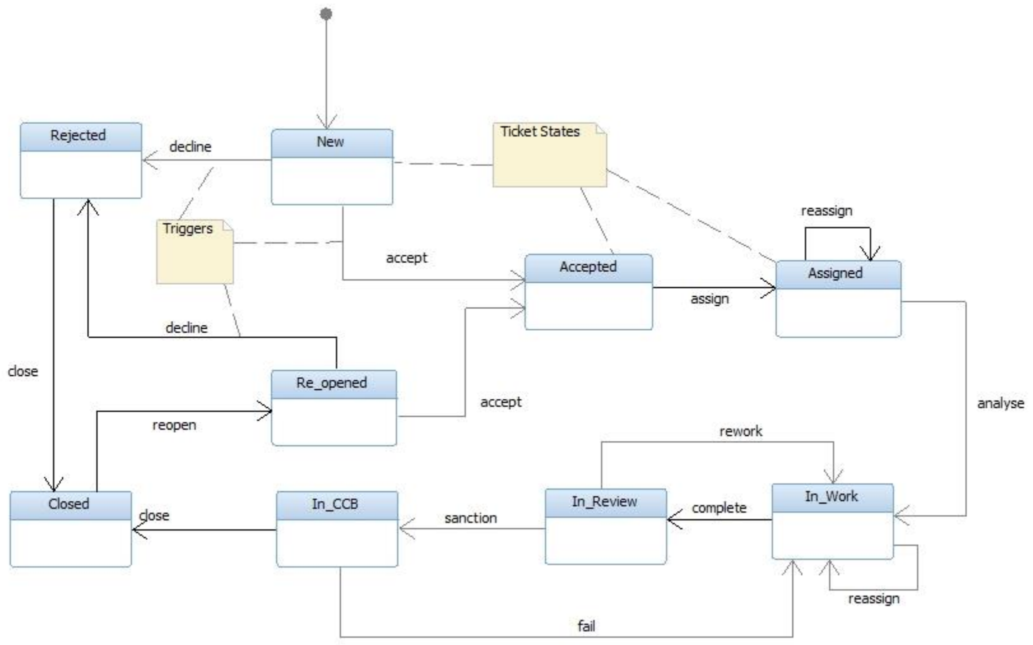


Figure 10: Ticket Workflow

State	State Description	Exit Transition	New State	Transition Description
New	A new ticket can be entered by anybody given access to the Trac database. The ticket starts in the new state from where it must be accepted or rejected.	accept	accepted	The ticket has been accepted as a valid change for which work should proceed.
		decline	rejected	The ticket has been rejected for a valid reason (i.e it is a duplicate of another ticket or requests a change that is not acceptable).

State	State Description	Exit Transition	New State	Transition Description
Accepted	The ticket has been accepted for work.	assign	assigned	The ticket has been assigned to a specific user. The assigned user will be the best placed resource for initial analysis of the requested change.
Rejected	The ticket has not been accepted for work.	close	closed	No further work is required on this ticket.
Assigned	Someone is now responsible for the ticket. It can be re-assigned or can be worked before progressing to the Review state.	reassign	assigned	The assigned user has allocated the ticket to a different user. This should only be done through the agreement of the two parties involved, and should not be a unilateral decision.
		analyse	in_work	The assigned user has performed initial analysis of the ticket and concluded that more detailed work needs to be performed before the ticket can be put forward for review.
In_Work	Work is underway to resolve the problem described by the ticket	complete	in_review	The assigned user has completed the required work to make the change but the change has not yet been reviewed locally before submission to the CCB.

State	State Description	Exit Transition	New State	Transition Description
		reassign	in_work	The assigned user has allocated the ticket to a different user. This should only be done through the agreement of the two parties involved, and should not be a unilateral decision.
In_Review	A member of the review team will check that the ticket has been satisfactorily resolved.	sanction	in_CCB	The local review has been completed satisfactorily and the change details can be submitted to the CCB.
		rework	in_work	The local review discovered one or more errors and the changes are passed back to the previous state for correction.
In_CCB	The ticket and proposed resolution will be examined by the NGVA CCB before deciding to close or reject the ticket.	close	closed	The CCB has reviewed the proposed changes and recommended that the changes are implemented and the ticket closed
		fail	in_work	The CCB has reviewed the proposed changes and recommended that the changes are reworked.
Closed	The ticket has been resolved.	reopen	re-opened	A decision has been taken to reopen a previous ticket.
Re-opened	A previously closed ticket can be re-opened if work related to its	accept	accepted	The ticket has been accepted as a valid change for which work should proceed.

State	State Description	Exit Transition	New State	Transition Description
	original context becomes necessary. The ticket re-enters the system and is either accepted or declined.	decline	rejected	The ticket has been rejected for a valid reason (i.e it is a duplicate of another ticket or requests a change that is not acceptable).

Table 1: Ticket Workflow State/Transition Descriptions

6.6. Ticket Closure

Closure of tickets will be sanctioned by the CCB using the following process:

1. CCB meetings shall dedicate time to progress existing tickets
2. Tickets to be discussed at a CCB meeting shall be identified in advance.
3. Where possible, consensus shall allow a ticket to be resolved before a meeting. This is likely to be possible for simple 'defect' type tickets.
4. The discussion at a meeting shall be based upon the opinion and solutions raised in the given ticket commentary - thus allowing the attendees to discuss the various solutions with colleagues before attending. In this way it should be possible to either:
   a. Resolve a ticket in the meeting.
   b. If necessary raise further issues for discussion via Trac to be resolved at the next meeting.
5. When no consensus regarding resolution can be found, tickets shall be resolved by majority decision.

CHAPTER 7 DDS AND QUALITY OF SERVICE

7.1. Overview

DDS Middleware allows the application of a Quality of Service (QoS) on a per topic basis, effectively allowing the system architect to tune the performance of the system based on the data received by applications.

QoS Policies can be applied to a number of DDS entity types although not all policies can be applied to all entity types. The entity types to which QoS policies can be applied are:

1. DomainParticipant - defines the scope of an application within a single, domain. It is a factory for creating other DDS entities in its domain.
2. Topic - allows publishers and subscribers to link with a common data type (data structure) by a Topic name. Where a Topic name is uniquely and unambiguously identified in a domain (e.g. “Heading_T”).
3. Publisher is the factory, container and manager for one or more heterogeneous (differently typed) DataWriters.
4. Subscriber - is the factory, container and manager for one or more heterogeneous (differently typed) DataReaders.
5. DataWriter - is strongly typed, and publishes data on a Topic. A DW can be configured to publish data synchronously, or asynchronously.
6. DataReader - is strongly typed, and subscribes to data on a Topic; it receives data published on a Topic.

7.2. Quality of Service Policies

QoS policies permit applications to manage, prioritize and shape data-flow in a network. Table 2 lists the Quality of Service policies and how they are supported by Topics, DataWriters & DataReaders.

	Participating DDS Entity		
QoS Policy	**Topic**	**DataWriter**	**DataReader**
Durability	Y	Y	Y
Durability Service	Y	Y	N
Deadline	Y	Y	Y
LatencyBudget	Y	Y	Y
Liveliness	Y	Y	Y
Reliability	Y	Y	Y
DestinationOrder	Y	Y	Y
History	Y	Y	Y
ResourceLimits	Y	Y	Y
TransportPriority	Y	Y	N
Lifespan	Y	Y	N
Ownership	Y	Y	Y
OwnershipStrength	N	Y	N
WriterDataLifecycle	Y	N	N
TimeBasedFilter	N	N	Y

Table 2: QoS Policies versus DDS Entities

Each of the QoS policies contained within Table 2 is discussed further in the following sections.

7.2.1. Durability

Durability controls whether or not new DataReaders get data which was written by DataWriters previously. The Durability can vary from not at all (the default) to persistent (stored to disk). This QoS policy helps insulate system from startup dependencies and can increase system tolerance to failure conditions.

7.2.2. Durability Service

When a DataWriter's Durability QoS is set to PERSISTENT_DURABILITY or TRANSIENT_DURABILITY, an external service is used to store and possibly forward the data sent by the DataWriter to DataReaders that are created after the data was initially sent.

7.2.3. Deadline

For DataReaders: deadline specifies the maximum expected elapsed time between arriving data samples. For DataWriters: deadline specifies a commitment to publish samples with no greater than this elapsed time between them.

7.2.4. LatencyBudget

This suggests how much time is allowed to deliver data. This is an optional QoS, the default value is 0, which implies you want to send with minimum latency.

7.2.5. Liveliness

This configures the mechanism that allows DataReaders to detect when matching DataWriters have become disconnected or dead. This can be used to ensure that important messages can be received if they are sent. For example, if a command message like “Emergency Stop” is never sent unless the situation is encountered, Liveliness can be used to periodically send a message to test the end-to-end connectivity of the system so that when an “Emergency Stop” message is sent, it will likely be received by all subscribers.

7.2.6. Reliability

This QoS policy turns on the Real Time Publish Subscriber (RTPS) reliability protocol between those DataWriters and DataReaders that set this QoS policy to the RELIABLE value. When the reliability protocol is used, DDS will attempt to repair samples that were not successfully received by reliable DataReaders. A connection between a DataWriter and DataReader may be configured for BEST_EFFORT Reliability, which means that no resources are used to monitor or guarantee that the data sent by a DataWriter is received by a DataReader. For periodic update of sensor values, "best effort" delivery is often good enough. It is the fastest, most efficient, and least resource-intensive method of getting the newest/latest value for a topic from DataWriters to DataReaders. However, there is no guarantee that the data sent will be received.

7.2.7. DestinationOrder

This controls how DDS deals with data sent by multiple DataWriters for the same topic. When multiple DataWriters send data for the same Topic, the order in which data from different DataWriters is received by the applications of different DataReaders may be different, hence DataReaders may not receive the same "last" value when DataWriters stop sending data. If DestinationOrder is set to "by reception timestamp", data will be delivered by a DataReader in the order in which it was received (this can lead to inconsistent final values). If DestinationOrder is set to "by source timestamp", data will be delivered by a DataReader in the order in which it was sent. If data arrives on the network with a source timestamp earlier than the source timestamp of the last data delivered, the new data will be dropped. This ordering therefore works best when system clocks are relatively synchronized among writing machines.

7.2.8. History

This controls how much data to store and how stored data is managed for a DataWriter or DataReader. Two settings are available: KEEP_ALL or KEEP_LAST with a value (depth). KEEP_ALL does not imply that DDS will store infinite data. How much data can actually be stored (and thus memory allocation) is controlled by the ResourceLimits QoS. When set to KEEP_LAST, the depth (i.e. the number of data samples to keep) applies on a per instance-basis (unique key value) for Topics that are keyed.

7.2.9. ResourceLimits

This controls amount of physical memory that is allocated for middleware entities; if dynamic allocations are allowed and how they occur; and memory usage among different instance values for keyed topics. ResourceLimits configures the amount of memory a DataWriter or DataReader may allocate to store data in a local cache (i.e. send or receive queues, respectively). The max_samples parameter in this policy has a role in throttling the send rate of reliable DataWriters, although using the “send window” properties of the DataWriterProtocol QoS policy is the recommended way to configure this behavior.

This QoS policy can limit how much system memory can be allocated by the middleware.

7.2.10. TransportPriority

This tells DDS that the data being sent has a different "priority" than other data. This provides a priority value to the underlying transport protocol, for those that can use it. Some transport protocols have a concept of user-settable “priorities” that may be used by operating system network stacks and switching hardware in between the publisher and the subscriber. For such transports and on supported Operating Systems (Oss), DDS will pass this value to the transport for its use. For other transports and other OSs, the middleware will ignore this value.

7.2.11. Lifespan

This specifies how long DDS should consider data sent by a user application to be valid. DDS will timestamp all data sent and received. When a finite Lifespan is specified for a DataWriter or DataReader, DDS will check to see how long the data has been stored in the DataWriter’s send queue or the DataReader’s receive queue and remove any data that has exceeded its Lifespan duration.

7.2.12. Ownership

This specifies if a DataReader can receive new samples for an instance of data from multiple DataWriters at the same time. By default, DataReaders for a given topic can receive data from any matching DataWriter for the same topic—this is the "shared" setting for the Ownership QoS policy. DataReaders can also be configured to use "exclusive" Ownership for a topic so that they only receive data from one DataWriter at a time. Ownership applies on a per key value (instance) basis for Topics that are keyed. Thus, with exclusive Ownership, DataReaders will only receive data from a single DataWriter for a particular instance (unique value for the key) of a Topic. This implies that a DataReader may receive data from multiple DataWriters as long as the DataWriters are sending data for different instances.

The value of Ownership must be the same for a DataWriter to be connected to a DataReader. Either both sides must be shared or both sides must be exclusive. Mismatched DataWriter and DataWriter pairs are not connected and will never exchange data.

7.2.13. OwnershipStrength

This specifies if a DataReader can receive new samples for an instance of data from multiple DataWriters at the same time.

The OwnershipStrength QoS policy is used to determine which DataWriter is allowed to send data (or updates for instances for keyed Topics) to DataReaders when Ownership is exclusive and there are multiple DataWriters all sending data for the same instance. The DataWriter with the highest value for the OwnershipStrength QoS policy will be considered the owner of the instance of the Topic and whose data is delivered to DataReaders. Data for the instance sent by all other DataWriters with lower OwnershipStrength will be dropped by DDS when received at the subscribing application.

7.2.14. WriterDataLifecycle

This controls how a DataWriter handles the lifecycle of the instances that it is registered to manage. This QoS policy applies on a per instance (key) basis for keyed Topics, so that when a DataWriter unregisters an instance, DDS can automatically also dispose that instance. This is the default behavior.

In cases where the ownership of a Topic is exclusive, DataWriters may want to relinquish ownership of a particular instance of the Topic to allow other DataWriters to send updates for the value of that instance regardless of how the OwnershipStrength QoS policy is set. In that case, you may only want a DataWriter to unregister an instance without disposing the instance. Disposing an instance is a statement that an instance no longer exists.

7.2.15. TimeBasedFilter

This sets a minimum time period before new data for an instance is provided to a DataReader, excess data sent faster than the period set are not sent or otherwise discarded. DataWriters may send data faster than needed by a DataReader. For example, a DataReader of sensor data that is displayed to a human in a GUI application often does not need data updates faster than a human can reasonably perceive changes in data values. This policy provides the ability to optimize resource usage by only delivering the required amount of data to different DataReaders and filtering out samples that arrive faster than a specified rate. Time-based data-filtering does not depend on any DataWriter settings. However, it also does not force the DataWriter to send at a specific rate. The update rate of new data is entirely under the control of the application code.

7.3. Design Patterns

Design patterns exist for different types of data within a system; typical patterns are contained within the following sections.

7.3.1. State Pattern

The State pattern models a set of information that describes the condition of a physical or logical object that can evolve and change over time. It differs from the event pattern in that an event notifies a set of recipients of the occurrence of a significant event. State data is produced by one or multiple writers to make it available for multiple readers interested in observing the concerned state evolution and react consequently.

State data is characterized by the following properties:

1. Asynchrony - State data producers are completely decoupled from their consumers. No Initial knowledge of the producer identity, its life cycle and data availability time is required. State data consumers can join the system at any time and should be able read data produced prior to their joining the system. No assumption is made regarding the amount of data to sustain.
2. Durability - State data should be kept available as long as it is needed. The durability property allows this data to be made available to late joining consumers.
3. Reliable delivery - The state pattern requires reliable state transfer such that a consistent view of the state can be maintained between all observers and the state producer. All the observers must have the last state update.

For details of the QoS Policies and settings that comprise this pattern see Annex A.1

7.3.2. Command Pattern

The command pattern models a set of information that is used to change the state of a physical or logical object on demand. It should be reliable but does not require an acknowledgement.

For details of the QoS Policies and settings that comprise this pattern see Annex A.2

7.3.3. Event Pattern

The Event pattern describes the ability to publish data structures that notify a set of recipients of a significant event. Each recipient could react differently to the occurrence of the event. Event based communication is effectively a decoupled communication between objects. The Event pattern can have many variants by setting additional QoS policies that are not imposed by the basic pattern, for example, a reliable Event Pattern can be set such that events are delivered in a reliable way. This is achieved by setting the Reliability policy to Reliable. Similarly, without the Durability policy set, events are considered to be volatile. Event data can be made non-volatile by setting the Durability policy to value to suit the application needs.

Event data is characterized by the following properties:

1. Asynchrony and anonymity - events suppliers and consumers are completely decoupled and unknown to each other. Suppliers can generate events without knowing the identities of the consumers. Conversely, consumers can receive events without knowing the identities of the suppliers.
2. Interest concept - recipients can choose to receive a subset of the all published events i.e. only those that is of interest to them.

For details of the QoS Policies and settings that comprise this pattern see Annex A.3

7.3.4. Alarm Pattern

The Alarm pattern handles the alert of a sudden event that must be processed as quickly as possible. The Alarm pattern is based on the Event pattern with an additional feature. Each alarm event requires a response event that handles the alarm and informs both the supplier and the other alarm consumers that the event is received and will be managed.

The Alarm pattern is applicable to event based systems whose event data is characterized by the following properties:

1. Exception and urgency - alarm events are exceptional events that signify an abnormal situation; one that must be processed as quickly as possible. Alarms are produced to report a special condition of the system that requires an urgent reaction. The urgency level of the alarm event depends on the criticality level of the reported situation and impacts the reaction deadline.
2. Reliable delivery and durability - the alarm event is too important to be lost. Every alarm event must be delivered reliably to each consumer and maintained until it is handled.
3. An acknowledgement message is required - the alarm event must be processed and the alarm event supplier needs to know that the event is effectively handled by someone.
4. Short durability - the alarm event is raised then cleared. It exists as long as the reported situation is not handled. When the return message is received by the supplier, the event must be cleared and becomes meaningless.

For details of the QoS Policies and settings that comprise this pattern see Annex A.4

7.3.5. Continuous Data (Periodic) Pattern

The Continuous Data pattern can be used to report information that is changing constantly. One or multiple producers publish the data to many consumers and update it periodically. The consumers may choose to not receive all the updates. A variant of the Continuous Data Pattern i.e. Continuous and Filtered Data can be defined where consumers choose to not receive all the produced data. This can be implemented by the TIME_BASED_FILTER QoS policy applied to the DataReaders. Consumers can specify a time duration that represents the minimum period separating two data updates at the consumer. This duration must be longer than the DEADLINE duration.

The Continuous Data pattern can be used to report information that is changing constantly. One or multiple producers publish the data to many consumers and update it periodically. The consumers may choose to not receive all the updates.

The Continuous data pattern is characterized by rapidly changing data - the exchanged data is periodically updated. It describes a changing state over time.

For details of the QoS Policies and settings that comprise this pattern see Annex A.5

7.3.6. Watchdog pattern

The Watchdog pattern allows an application to report information about its liveliness. Liveliness information can be viewed as a structured set of data describing the activity status of each functional part of the application. This data is periodically updated and disseminated to make it available to interested consumers. This pattern is applicable to highly available systems that must provide their services continuously. The system software and hardware resources are closely monitored to check their liveliness status and manage any resource “death” as quickly as possible.The pattern applications characteristics may be summarized as follows:

1. Data describes life status of the processing units making the application that need to be monitored.
2. Best Effort Data delivery
3. Periodic updating of the data
4. Short data validity duration - liveliness data is short-lived data whose instances expire very quickly because it is supposed to report the real and current life status of the concerned resource instantaneously.

The Watchdog pattern is similar to the State pattern but remains unique. The Watchdog reports the liveliness status of monitored resources to a set of resource monitors, operating in the observer role, and the state data represents the liveliness data. The difference between the watchdog and State patterns is that liveliness data is not durable and only the last value is important. A resource monitor is interested in only the latest and last value of the data without caring about past data values that have in fact expired. This pattern is also a specialization of the Continuous Data Pattern. In fact, the watchdog role is a special continuous data supplier, the resource monitor is a special continuous data consumer and the continuous data corresponds to the liveliness data.

For details of the QoS Policies and settings that comprise this pattern see Annex A.6

7.3.7. Configuration/Specification pattern

Most applications have configuration parameters that are either set at start-up or changed during the execution of the application to better adapt it to some new conditions. The use of DDS for setting and changing these parameters allows for a great flexibility. The Configuration/Specification topic is persistent so that they are available everywhere at start-up, in a fault tolerant manner. They can be changed from anywhere else (even offline) using ephemeral writers or some kind of console tool.

The pattern allows for a dynamic view of configuration that can be updated and performed at any time, not only at start-up.

For details of the QoS Policies and settings that comprise this pattern see Annex A.7

CHAPTER 8 MODELLING CONVENTIONS

8.1. Platform Independent Models

8.1.1. Use Cases

A Use Case diagram, that captures the capabilities of the domain under consideration, shall be constructed for each modeling domain. The Use Case diagram shall be constructed using IBM Rational Rhapsody and will form part of the Rhapsody file for that module.

8.1.2. Class Diagram

A Class diagram shall be constructed from the Use Case Diagram for the domain under consideration

The Class diagram or PIM is a Unified Modeling Language (UML) static structure diagram that describes the structure of a module by showing the module's classes, their attributes, operations, and the relationships between classes.

A Class diagram shall be constructed for each module using IBM Rational Rhapsody and will form part of the Rhapsody file for that module.

8.1.3. Naming Rules

The following naming rules shall be obeyed for each class:

1. The name given to each class shall be of the form Xxx_Yyy_Zzz or XxxYyyZzz (e.g. Navigation_Resource_Specification or NavigationResourceSpecification);

2. Each Class may contain a number of attributes. The name given to each attribute shall use headlessCamelCase (e.g maximumDataCalculationRate);

3. Each Class may contain a number of operations. The name given to each operation shall use headlessCamelCase (e.g setNorthReference);

4. Each attribute of a class shall be assigned a data type, which may take the form of:

    a. A class
    b. A structure
    c. An enumeration
    d. A typedef

5. Each structure shall be a Rhapsody Type of Kind = Structure. The name given to each structure shall be of the form XxxYyyZzzType (e.g. LinearVelocity2DType);

6. The name given to attributes of a Structure shall use headlessCamelCase (e.g heading).

7. Each enumeration shall be a Rhapsody Type of Kind = Enumeration. The name given to each enumeration shall be of the form XxxYyyZzzType;

8. The name given to literals of an enumeration shall be of the form XXX_YYY_ZZZ_TYPE__AAA (e.g. COORDINATE_SYSTEM_TYPE__BNG & COORDINATE_SYSTEM_TYPE__MGRS are two literals of an enumeration named CoordinateSystemType);

9. Each typedef shall be a Rhapsody Type of Kind = Typedef. The name given to each typedef shall be of the form ParameterInUnitsType (e.g CurrentInAmpsType & FlowrateInCubicMetresPerSecType are two typedefs defined in the Common Module).

CHAPTER 9 MODEL TRANSLATIONS

9.1. Translation of PIM to IDL

The translation of the PIM to IDL is carried out by two Translators:

1. PIM to PSM Translation
2. PSM to IDL Translation

The translators can be invoked independently or the entire process can be translated from a single command.

9.1.1. PIM to PSM translation

The PIM Translator is designed to be used in conjunction with IBM Rational Rhapsody and should be installed in accordance with NGVA PIM Translator User Manual Version 1.1, Sections 2 & 3[6]. The NGVA PIM Translator will not function with Enterprise Architect

When invoked, the NGVA PIM Translator will translate the classes of a PIM and create a new PSM package.

It is important to note that Rhapsody properties are part of a containment structure (see Figure 11). Setting properties at the Class level will result in those properties being set at all lower levels so it is important to invoke the correct properties dialog for the appropriate level when setting properties for PIM to PSM translation.

The NGVA PIM Translator uses properties associated with Metaclasses as shown in Figure 12. Each Metaclass has a number of properties that can be set prior to translation, for each instance in the PIM

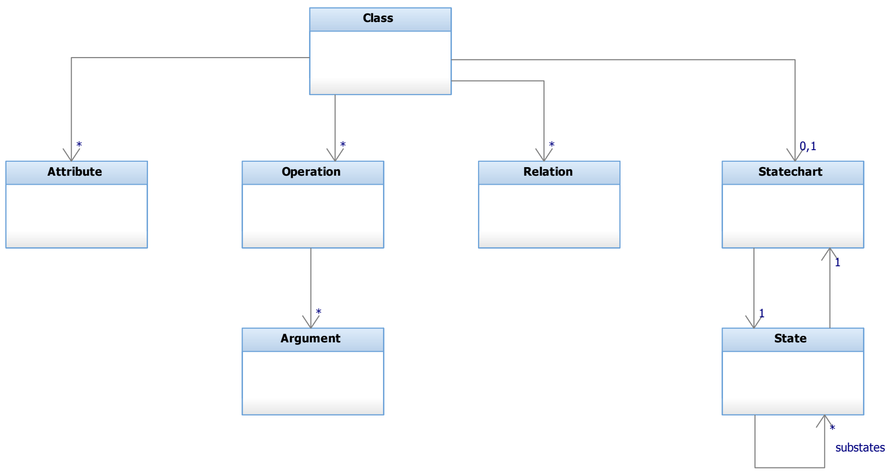
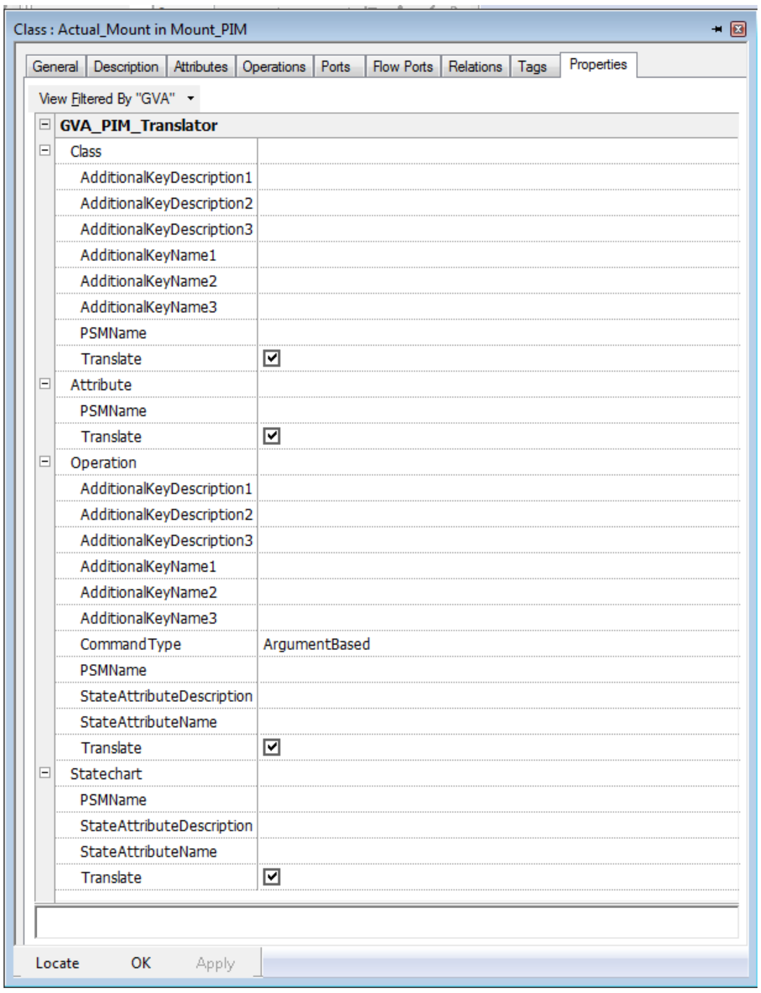


Figure 11: Rhapsody Containment Hierarchy

[6] The documentation for the NGVA PIM & PSM Translators will be made available via the Data Model Repository.

To set the properties of a PIM Class at the Class Level the Metaclass Properties dialog box is invoked by:

1. selecting a PIM class in the Rhapsody browser
2. right clicking on that selection
3. selecting Features from the context menu that appears
4. Setting the Filter on the dialog box that appears to ‘GVA’ and ‘Match category name’
5. Selecting the Properties tab



GVA_PIM_Translator	
Class	
AdditionalKeyDescription1	
AdditionalKeyDescription2	
AdditionalKeyDescription3	
AdditionalKeyName1	
AdditionalKeyName2	
AdditionalKeyName3	
PSMName	
Translate	✓
Attribute	
PSMName	
Translate	✓
Operation	
AdditionalKeyDescription1	
AdditionalKeyDescription2	
AdditionalKeyDescription3	
AdditionalKeyName1	
AdditionalKeyName2	
AdditionalKeyName3	
CommandType	ArgumentBased
PSMName	
StateAttributeDescription	
StateAttributeName	
Translate	✓
Statechart	
PSMName	
StateAttributeDescription	
StateAttributeName	
Translate	✓

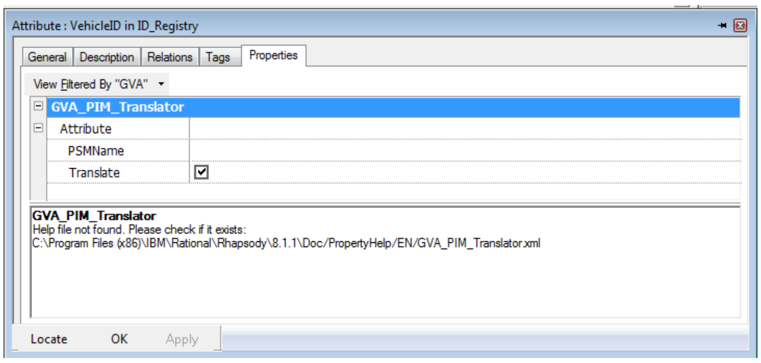
Figure 12: Metaclass Properties of a Class

The setting of the properties at this level is described in the NGVA PIM Translator Requirements Specification Version 2.3 Section 3.4

To set the properties of an Attribute of a PIM Class the Metaclass dialog box for an Attribute of a Class is invoked by:

1. selecting the attribute of a PIM class in the Rhapsody browser
2. right clicking on that selection
3. selecting Features from the context menu that appears

4. Setting the Filter on the dialog box that appears to ‘GVA’ and ‘Match category name’
5. Selecting the Properties tab



Attribute : VehicleID in ID_Registry

General | Description | Relations | Tags | Properties

View Filtered By "GVA"

GVA_PIM_Translator	
Attribute	
PSMName	
Translate	☑

GVA_PIM_Translator
Help file not found. Please check if it exists:
C:\Program Files (x86)\IBM\Rational\Rhapsody\8.1.1\Doc/PropertyHelp/EN/GVA_PIM_Translator.xml

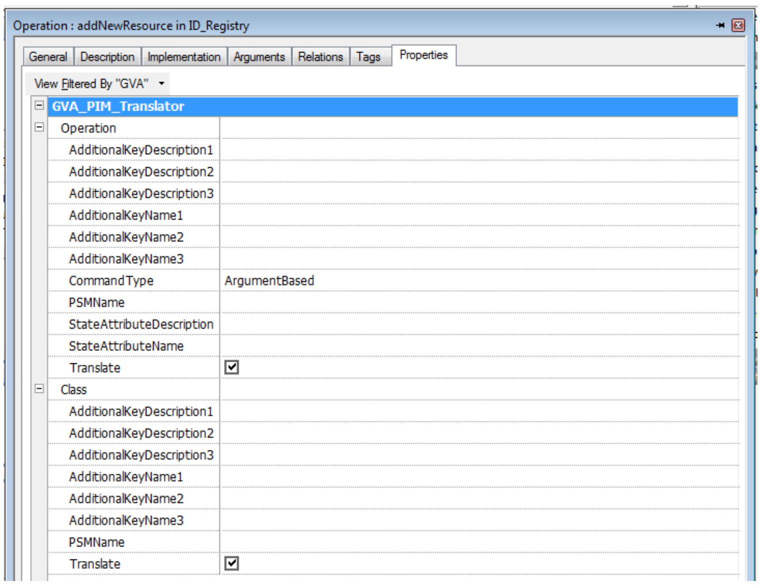
Locate OK Apply

Figure 13: Metaclass Properties of a Class Attribute

The setting of the properties at this level is described in the NGVA PIM Translator Requirements Specification Version 2.3 Section 3.5.

To set the properties of an Operation of a PIM Class the Metaclass dialog box for an Operation of a Class is invoked by:

1. selecting the operation of a PIM class in the Rhapsody browser
2. right clicking on that selection
3. selecting Features from the context menu that appears
4. Setting the Filter on the dialog box that appears to ‘GVA’ and ‘Match category name’
5. Selecting the Properties tab



Operation : addNewResource in ID_Registry

General | Description | Implementation | Arguments | Relations | Tags | Properties

View Filtered By "GVA"

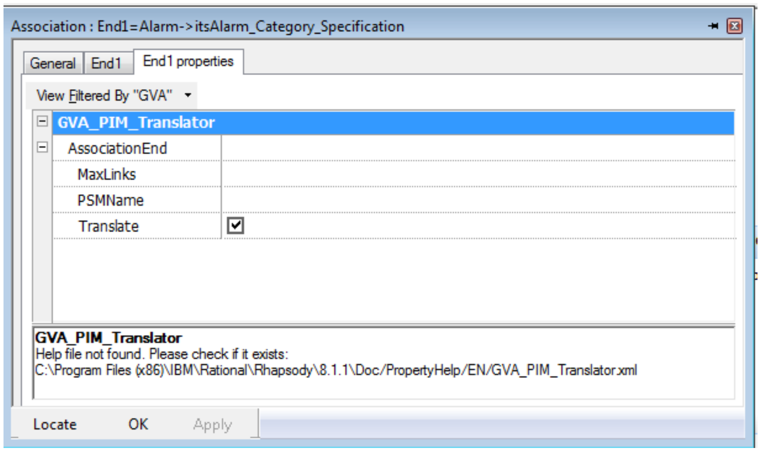
GVA_PIM_Translator	
Operation	
AdditionalKeyDescription1	
AdditionalKeyDescription2	
AdditionalKeyDescription3	
AdditionalKeyName1	
AdditionalKeyName2	
AdditionalKeyName3	
CommandType	ArgumentBased
PSMName	
StateAttributeDescription	
StateAttributeName	
Translate	☑
Class	
AdditionalKeyDescription1	
AdditionalKeyDescription2	
AdditionalKeyDescription3	
AdditionalKeyName1	
AdditionalKeyName2	
AdditionalKeyName3	
PSMName	
Translate	☑

Figure 14: Metaclass Properties of a Class Operation

The setting of the properties at this level is described in the NGVA PIM Translator Requirements Specification Version 2.3, Section 3.9.

To set the properties of a Relation of a PIM Class the Metaclass Properties dialog box for a Relation between two Classes is invoked by:

1. selecting the relation between PIM classes on the Class diagram
2. right clicking on that selection
3. selecting Features from the context menu that appears
4. Setting the Filter on the dialog box that appears to ‘GVA’ and ‘Match category name’
5. Selecting the Properties tab



Association : End1=Alarm->itsAlarm_Category_Specification

View Filtered By "GVA"

GVA_PIM_Translator	
AssociationEnd	
MaxLinks	
PSMName	
Translate	☑

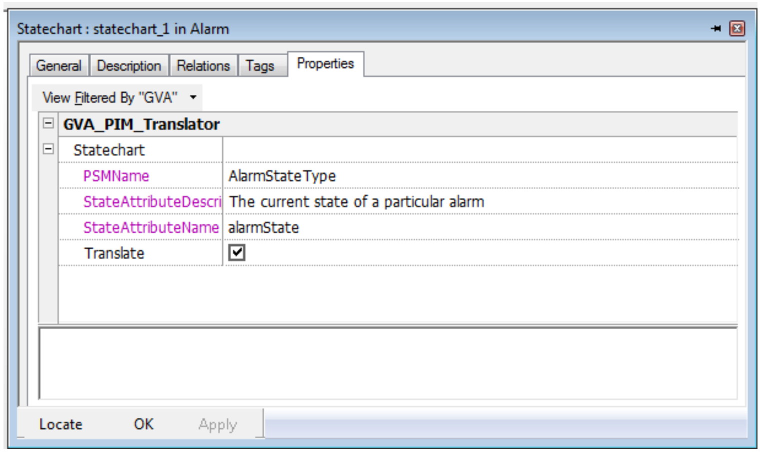
GVA_PIM_Translator
Help file not found. Please check if it exists:
C:\Program Files (x86)\IBM\Rational\Rhapsody\8.1.1\Doc/PropertyHelp/EN/GVA_PIM_Translator.xml

Figure 15: Metaclass Properties of a Relation

The setting of the properties at this level is described in the NGVA PIM Translator Requirements Specification Version 2.3, Section 3.9.

Setting the properties of a Statechart attached to a PIM Class the Metaclass Properties dialog box for a Statechart attached to a Class is invoked by:

1. selecting the statechart of a PIM class in the Rhapsody browser
2. right clicking on that selection
3. selecting Features from the context menu that appears
4. Setting the Filter on the dialog box that appears to ‘GVA’ and ‘Match category name’
5. Selecting the Properties tab



GVA_PIM_Translator	
Statechart	
PSMName	AlarmStateType
StateAttributeDescri	The current state of a particular alarm
StateAttributeName	alarmState
Translate	✓

Figure 16: Metaclass Properties of a Statechart

The setting of the properties at this level is described in the NGVA PIM Translator Requirements Specification Version 2.3, Section 3.7.

9.1.2. PSM to IDL Translation

The NGVA PSM Translator is designed to be used in conjunction with IBM Rational Rhapsody and should be installed in accordance with NGVA PSM Translator User Manual Version 1.1, Sections 2 & 3. The NGVA PSM Translator will not function with Enterprise Architect.

When invoked, the NGVA PSM Translator will translate the classes of a PSM and to a set of IDL text files.

Given that the following conditions are true, the NGVA PSM Translator will produce IDL capable of being compiled by DDS vendor's tools:

1. That the NGVA PIM Translator has produced zero errors
2. That the NGVA PSM Translator has been correctly configured according to NGVA PSM Translator User Manual Version 1.1, Section 3.

INTENTIONALLY BLANK

ANNEX A QOS PATTERNS

A.1. STATE PATTERN

QoS Policy	Possible Values	Concerned DDS Entities	Default Values	Remarks
Durability	TRANSIENT_LOCAL TRANSIENT PERSISTENT	Topic DataReader DataWriter	TRANSIENT_LOCAL	The State pattern requires the durability value to be greater than or equal to TRANSIENT_LOCAL. Hence, possible values are TRANSIENT_LOCAL, TRANSIENT or PERSISTENT depending on the pattern application needs. The Durability policy must be attached to the state Topic, DataReaders and DataWriters. TRANSIENT_LOCAL fits applications where a state data type is always produced by the same DataWriter. If this DataWriter disappears, that state data is no longer produced. This means that the state data life cycle is tightly coupled to the DataWriter life cycle. A DataReader that joins after the DataWriter disappears will no longer have access to the produced state information. TRANSIENT fits applications whose state data outlives its producer life cycle. A DataReader that joins after the DataWriter disappears can read the produced state data as long as the system is operational. This means that the state data life cycle is tightly coupled to the whole system life cycle. PERSISTENT fits applications whose state data outlives the system life cycle and needs to be kept on a permanent storage (ex: file, data base, ...), so that if the system is stopped and restarted the persistent state data is once more made available.
Reliability	Reliable	Topic DataReader DataWriter	Reliable	This QoS policy indicates the level of reliability of data delivery. Since the State pattern requires reliable state data delivery, this policy value must be set to RELIABLE. The Reliability policy must be attached to the state Topic, DataReaders and DataWriters. Fits applications where a state data type is always produced by the same DataWriter. If this DataWriter disappears, that state data is no longer produced. This means that the state data life cycle is tightly coupled to the DataWriter life cycle. A DataReader that joins after the DataWriter disappears will no longer have access to the produced state information.

QoS Policy	Possible Values	Concerned DDS Entities	Default Values	Remarks
DestinationOrder	By_Source_Timestamp By_Destination_Tmestamp	Topic DataReader DataWriter	By_Source_Timestamp	This policy controls how each subscriber resolves the final value of a data instance that is written by multiple DataWriter objects (which may be associated with different Publisher objects) running on different nodes. It indicates the determination of the logical order among changes made by the Publisher to the same topic instance. The pattern imposes the By_Source_Timestamp value in order to guarantee that state changes will be seen in the same order by all the observers. Hence, data is ordered according to a timestamp placed at the source. This enforces state data consistency since transport times cannot corrupt the ordering of received data.
History	Keep_Last Keep_All	Topic DataReader DataWriter	Keep_Last	This QoS policy controls the behaviour of the DDS infrastructure when the data changes before it is delivered to all DataReaders. The policy indicates how much data will be available. The State pattern imposes a Keep_Last value with an application defined depth. The application must determine the history depth of state data it requires to make available to DataReaders.
WriterDataLlifecycle	autodispose_unregistered_instances =FALSE	DataWriter	autodispose_unregistered_instances =FALSE	This policy controls the life cycle of data instances with regard to its DataWriter life cycle. The policy specifies whether State Data instances are available if the associated DataWriter becomes unregistered or deleted. The pattern imposes the setting of the “autodispose_unregistered_instances” flag to FALSE in order to keep the state data completely independent from the life cycle of its DataWriter.

A.2. COMMAND PATTERN

QoS Policy	**Possible Values**	**Concerned DDS Entities**	**Default Values**	**Remarks**
Durability	VOLATILE	Topic DataReader DataWriter	VOLATILE	The Command pattern requires the durability value to be set to Volatile. A Command should be sent only once hence there is no need for any persistence.
Reliability	Reliable	Topic DataReader DataWriter	Reliable	This QoS policy indicates the level of reliability of data delivery. Since the pattern requires reliable data delivery, this policy value must be set to RELIABLE. The Reliability policy must be attached to the Topic, DataReaders and DataWriters.
History	KEEP_ALL	Topic DataReader	KEEP_ALL	This QoS policy controls the behaviour of the DDS infrastructure when the data changes before it is delivered to all DataReaders. The policy indicates how much command data will be available..
Lifespan	2 Seconds	Topic DataReader DataWriter	2 Seconds	Liveliness data is ephemeral data that loses its validity over time because the associated resource life status can change at any time. The most valid and significant data instance is the last and the latest one. It is up to the application to estimate and specify realistic lifespan duration depending on the deadline period and the network latency.

A.3. EVENT PATTERN

QoS Policy	Possible Values	Concerned DDS Entities	Default Values	Remarks
WriterDataLlifecycle	autodispose_unregistered_instances =FALSE	DataWriter	autodispose_unregistered_instances =FALSE	This policy controls the life cycle of data instances with regard to its DataWriter life cycle. The policy specifies whether Data instances are available if the associated DataWriter becomes unregistered or deleted. The pattern imposes the setting of the “autodispose_unregistered_instances” flag to FALSE in order to keep the data completely independent from the life cycle of its DataWriter.
HISTORY	KEEP_LAST	Topic Datareader Datawriter	KEEP_LAST	This QoS policy controls the behaviour of the DDS infrastructure when the data changes before it is delivered to all DataReaders. The policy indicates how much data will be available. The Event pattern imposes a KEEP_LAST value with an application defined depth. The application must determine the history depth of event data it requires to make available to DataReaders.

A.4. ALARM PATTERN

QoS Policy	Possible Values	Concerned DDS Entities	Default Values	Remarks
WriterDataLlifecycle	autodispose_unregistered_instances =FALSE	DataWriter	autodispose_unregistered_instances =FALSE	This policy controls the life cycle of data instances with regard to its DataWriter life cycle. The policy specifies whether Data instances are available if the associated DataWriter becomes unregistered or deleted. The pattern imposes the setting of the “autodispose_unregistered_instances” flag to FALSE in order to keep the data completely independent from the life cycle of its DataWriter.
DestinationOrder	By_Source_Timestamp By_Destination_Tmestamp	Topic DataReader DataWriter	By_Source_Timestamp	This policy controls how each subscriber resolves the final value of a data instance that is written by multiple DataWriter objects (which may be associated with different Publisher objects) running on different nodes. It indicates the determination of the logical order among changes made by the Publisher to the same topic instance. The pattern imposes the By_Source_Timestamp value in order to guarantee that state changes will be seen in the same order by all the observers. Hence, data is ordered according to a timestamp placed at the source. This enforces state data consistency since transport times cannot corrupt the ordering of received data.
Reliability	Reliable	Topic DataReader DataWriter	Reliable	This QoS policy indicates the level of reliability of data delivery. Since the State pattern requires reliable state data delivery, this policy value must be set to RELIABLE. The Reliability policy must be attached to the state Topic, DataReaders and DataWriters.

QoS Policy	**Possible Values**	**Concerned DDS Entities**	**Default Values**	**Remarks**
Durability	TRANSIENT_LOCAL TRANSIENT PERSISTENT	Topic DataReader DataWriter	TRANSIENT_LOCAL	The pattern requires the durability value to be greater than or equal to TRANSIENT_LOCAL. Hence, possible values are TRANSIENT_LOCAL, TRANSIENT or PERSISTENT depending on the pattern application needs. The Durability policy must be attached to the Topic, DataReaders and DataWriters. TRANSIENT_LOCAL fits applications where a state data type is always produced by the same DataWriter. If this DataWriter disappears, that data is no longer produced. This means that the data life cycle is tightly coupled to the DataWriter life cycle. A DataReader that joins after the DataWriter disappears will no longer have access to the produced information. TRANSIENT fits applications whose data outlives its producer life cycle. A DataReader that joins after the DataWriter disappears can read the produced data as long as the system is operational. This means that the data life cycle is tightly coupled to the whole system life cycle. PERSISTENT fits applications whose data outlives the system life cycle and needs to be kept on a permanent storage (ex: file, data base, ...), so that if the system is stopped and restarted the persistent data is once more made available.
History	KEEP_ALL	Topic DataReader	KEEP_ALL	This QoS policy controls the behaviour of the DDS infrastructure when the data changes before it is delivered to all DataReaders. The policy indicates how much data will be available. The Alarm pattern imposes a KEEP_ALL value with an application defined depth. The application must determine the history depth of event data it requires to make available to DataReaders.

A.5. CONTINUOUS DATA (PERIODIC) PATTERN

QoS Policy	Possible Values	Concerned DDS Entities	Default Values	Remarks
WriterDataLlifecycle	autodispose_unregistered_instances =FALSE	DataWriter	autodispose_unregistered_instances =FALSE	This policy controls the life cycle of data instances with regard to its DataWriter life cycle. The policy specifies whether Data instances are available if the associated DataWriter becomes unregistered or deleted. The pattern imposes the setting of the "autodispose_unregistered_instances" flag to FALSE in order to keep the data completely independent from the life cycle of its DataWriter.
DestinationOrder	By_Source_Timestamp By_Destination_Tmestamp	Topic DataReader DataWriter	By_Source_Timestamp	This policy controls how each subscriber resolves the final value of a data instance that is written by multiple DataWriter objects (which may be associated with different Publisher objects) running on different nodes. It indicates the determination of the logical order among changes made by the Publisher to the same topic instance. The pattern imposes the By_Source_Timestamp value in order to guarantee that changes will be seen in the same order by all the observers. Hence, data is ordered according to a timestamp placed at the source. This enforces data consistency since transport times cannot corrupt the ordering of received data.
RELIABILITY	BEST_EFFORT	Topic DataReader DataWriter	BEST_EFFORT	This QoS policy indicates the level of reliability of data delivery. Since the Continuos Data pattern is one that issues data on a periodic basis there is no requirement reliable data delivery, this policy value is therefore set to BEST_EFFORT. The Reliability policy must be attached to the Topic, DataReaders and DataWriters.
DEADLINE		Topic DataWriter		This QoS policy is required to guarantee that data is updated periodically. The associated period is Application dependant, hence, the Supplier commits to write a new value at least once every deadline period and the Consumer expects a new value at least once every deadline period.

QoS Policy	Possible Values	Concerned DDS Entities	Default Values	Remarks
LATENCY		Topic DataReader DataWriter	Zero	This is an optional QoS, the default value is 0, which implies you want to send with minimum latency.

A.6. WATCHDOG PATTERN

QoS Policy	Possible Values	Concerned DDS Entities	Default Values	Remarks
RELIABILITY	RELIABLE	Topic DataReader DataWriter	RELIABLE	This QoS policy indicates the level of reliability of data delivery. Since the pattern requires reliable data delivery, this policy value must be set to RELIABLE. The Reliability policy must be attached to the Topic, DataReaders and DataWriters.
DestinationOrder	By_Source_Timestamp	Topic DataReader DataWriter	By_Source_Timestamp	This policy controls how each subscriber resolves the final value of a data instance that is written by multiple DataWriter objects (which may be associated with different Publisher objects) running on different nodes. It indicates the determination of the logical order among changes made by the Publisher to the same topic instance. The pattern imposes the By_Source_Timestamp value in order to guarantee that changes will be seen in the same order by all the observers. Hence, data is ordered according to a timestamp placed at the source. This enforces data consistency since transport times cannot corrupt the ordering of received data.
LIFESPAN		Topic DataWriter		Liveliness data is ephemeral data that loses its validity over time because the associated resource life status can change at any time. The most valid and significant data instance is the last and the latest one. It is up to the application to estimate and specify realistic lifespan duration depending on the deadline period and the network latency.
DEADLINE		Topic DataReader DataWriter		The watchdog produces liveliness data periodically with a known period. This QoS policy is imposed to guarantee that behaviour for the watchdog. Based on this period, the deadline period must be specified by the application.

A.7. CONFIGURATION/SPECIFICATION PATTERN

QoS Policy	Possible Values	Concerned Entities	Default Values	Remarks
WriterDataLlifecycle	autodispose_unregistered_instances =FALSE	DataWriter	autodispose_unregistered_instances =FALSE	This policy controls the life cycle of data instances with regard to its DataWriter life cycle. The policy specifies whether Data instances are available if the associated DataWriter becomes unregistered or deleted. The pattern imposes the setting of the “autodispose_unregistered_instances” flag to FALSE in order to keep the data completely independent from the life cycle of its DataWriter.
Reliability	RELIABLE	Topic DataReader DataWriter	RELIABLE	This QoS policy indicates the level of reliability of data delivery. As the pattern imposes a reliable data delivery, this policy value is set to RELIABLE. It must be attached to the configuration Topic, and inherited by DataReaders and DataWriters.
Durability	PERSISTENT	Topic DataReader DataWriter	PERSISTENT	This QoS policy controls whether the DDS infrastructure will make data available to late joining readers. This QoS policy value must be PERSISTENT to ensure that configuration can be edited offline and that the latest modification remains available at any time. The value of this QoS value is imposed by the pattern and must never be changed when initializing the readers or writers.
History	KEEP_LAST Length=1	Topic DataReader	KEEP_LAST Length=1	This QoS policy controls the behaviour of the DDS infrastructure when the data changes before it is delivered to all DataReaders. For obvious reasons, the pattern imposes the KEEP_LAST value with a depth of 1. There is no need to make available more than one sample for such configuration information.

ANNEX B MODULE MATURITY LEVELS

Level	MML Description	Equivalent TRL Description
MML 9	Data interface model generated from module PIM has been embedded in multiple deployed designs and proven to operate.	Actual Technology system qualified through successful mission operations.
MML 8	Data interface model generated from module PIM has been embedded within an actual electronic architecture design which has passed all test and validation and is proven in-Service.	Actual technology system completed and qualified through test and demonstration.
MML 7	Data interface model generated from module PIM has been embedded within an actual electronic architecture design, and is ready for final test and demonstration.	Technology system prototype demonstration in an operational environment.
MML 6	Data interface model generated from module PIM has been embedded and implemented in a whole system context either on a systems integration rig or on an actual system using a majority of real components.	Technology system/subsystem model or prototype demonstration in a relevant environment.

Level	MML Description	Equivalent TRL Description
MML 5	Data interface model generated from module PIM has undergone testing of the complete set of classes for that PIM on a development rig which includes the simulation of operating applications that use the Interface data structures.	Technology component and/or basic technology subsystem validation in relevant environment.
MML 4	Data interface model generated from module PIM has undergone initial lab tests by ensuring that all classes have been exercised by at least one write operation and at least one read operation, thereby demonstrating correct Data transport.	Technology component and/or basic technology subsystem validation in laboratory
MML 3	Module PIM has been subject to several reviews, agreed by an approved review body or working group and has been translated and compiled without errors.	Analytical and experimental critical function and/or characteristic proof-of concept.

Level	MML Description	Equivalent TRL Description
MML 2	Module PIM has undergone a single review against relevant Use Cases at a stakeholder workshop session, module elements are fully documented and the module is compliant with LDM Methodology.	Technology concept and/or application formulated.
MML 1	Initial Use Cases and PIM created for Module.	Basic principles observed and reported

ANNEX C ABBREVIATIONS

CCB	Change Control Board
CI	Configuration Item
CM	Configuration Management
CPU	Central Processing Unit
CR	Compulsory Requirement
DDS	Data Distribution Service
DDSI	Data Distribution Service Interoperability
Def Stan	Defence Standard
EA	Enterprise Architect
GBA	Generic Base Architecture
GSA	Generic Soldier Architecture
GVA	Generic Vehicle Architecture
HUMS	Health & Usage Monitoring System
IBM	International Business Machines
IDL	Interface Design Language
ITT	Invitation to Tender
MDA	Model Driven Architecture
MilVA	Military Vetronics Association
MOD	Ministry of Defence
MQTT	Message Queue Telemetry Transport
NAAG	Nato Army Armaments Group
NATO	North Atlantic Treaty Organization
NGVA	NATO Generic Vehicle Architecture
NSA	NATO Standardisation Agency
OE	Optional Enhancement
OMG	Object Management Group
PIM	Platform Independent Model
PSM	Platform Specific Model
QoS	Quality of Service
RTPS	Real Time Publish Subscribe
SCM	Software Configuration Management
SS	System Specific
STANAG	Standardisation Agreement
UML	Unified Modelling Language
VCS	Version Control System
XMI	XML Metadata Interchange
XML	Extensible Markup Language

AEP-4754(A)(1)
VOL V