Date: June ~~2019~~2020

TACSIT data Exchange (TEX)

Version 1.0 - FTF beta 2

OMG Document Number dtc/~~19~~20-06-0~~2~~3

Normative Reference: https://www.omg.org/spec/TEX/1.0

Associated Normative Machine Consumable Files:

https://www.omg.org/spec/TEX/~~20190501~~20200520/tex.xmi
https://www.omg.org/spec/TEX/20180618/ts/tacsit-tex-datainterfac.d.ts
https://www.omg.org/spec/TEX/20180618/ts/tacsit-tex-datapayload.d.ts
https://www.omg.org/spec/TEX/20180618/dds/DataPayload_DDS_PSM.zip
https://www.omg.org/spec/TEX/20180618/csharp/DataPayload_CS_PSM.zip
https://www.omg.org/spec/TEX/20180618/csharp/DataSink_CS_PSM.zip
https://www.omg.org/spec/TEX/20180618/nvgtex/EntityChangeSinkEvent.xsd
https://www.omg.org/spec/TEX/20180618/nvgtex/EntityHistoryPayload.xsd
https://www.omg.org/spec/TEX/20180618/nvgtex/GroupChangeEvent.xsd
https://www.omg.org/spec/TEX/20180618/nvgtex/GroupChangeSinkEvent.xsd
https://www.omg.org/spec/TEX/20180618/nvgtex/GroupMetaData.xsd
https://www.omg.org/spec/TEX/20180618/nvgtex/HistoryChangeEvent.xsd
https://www.omg.org/spec/TEX/20180618/nvgtex/EntityChangeEvent.xsd
https://www.omg.org/spec/TEX/20180618/TCI/TCI_DDS_PSM.zip
https://www.omg.org/spec/TEX/20180618/TCI/TCI_CS_PSM.zip
https://www.omg.org/spec/TEX/20180618/TCI/TCI/tacsit-tci-controller.d.ts

This OMG document replaces the submission document (c4i/18-05-02, Alpha). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome, and should be directed to issues@omg.org by March 1, 2019.

You may view the pending issues for this specification from the OMG revision issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in June 2019. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

USE OF SPECIFICATION – TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents

for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

DISCLAIMER OF WARRANTY

RESTRICTED RIGHTS LEGEND

TRADEMARKS

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue.

Table of Contents

Table of Figures

Table of Tables

Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language®); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel™); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at:

http://www.omg.org/spec

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
109 Highland Avenue
Needham, MA 02494
USA

Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: pubs@omg.org

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification via the report form at:

http://issues.omg.org/issues/create-new-issue

This page intentionally left blank.

0 Scope

The domain of military and operational Command and Control systems is characterized by a huge variety of underlying computing platforms. A standard-based interface to common TACtical SITuation (TACSIT) services is thus essential for interoperable and open systems. In this scope, a first standard has already been issued by the OMG: the TACSIT Controller Interface consists of set of interfaces to interact with TACSIT display systems.

This specification offers data interfaces to TACSIT systems. More specifically, it standardizes:

- The data that need to be exchanged with TACSIT systems;
- The interfaces needed to exchange data from/to TACSIT systems.

1 Conformance

Implementations is considered to be in conformance to this standard if:

- It implements at least one the of the Data Payload PSM (see Clause 10);
- And it implements at least one the of the Data Interface PSM (see Clause 11) and/or at least one of the Data Sink PSM (see Clause 12) matching the preceding Data Payload PSM.

Implementing one of the Data Payload PSM shall implement the optional points as specified in Clause 6.2.5 and may include extensions as specified in Clause 6.2.6.

The conditions under which a Data Payload PSM matches a Data Interface PSM or a Data Sink PSM are specified in the specification of the Data Interface PSM (see Clause 11) or of the Data Sink PSM (see Clause 12).

Note: Implementing this standard does not entail to implement OMG's TCI standard even if there is a natural synergy between them.

2 Normative References

- TACSIT Controller Interface (**TCI**) 1.0 (formal/13-02-04) targets at the Controller services necessary for TACSIT software component manipulation only including such functionalities as range scaling, setting the area center/offset, setting view rotation, selecting objects, registering for events, and receiving cursor location updates;
- The Open Architecture Radar Interface Standard (**OARIS**) 1.0 (formal/16-03-02) primarily defines the interface between the CMS and a Radar system within a modular combat system architecture for naval platforms.
- The NATO Vector Graphics (**NVG**) protocol intends to provide military systems developers with a simple, yet extensible, specification for encoding battle-space information to support geospatial viewing (https://tide.act.nato.int/tidepedia/index.php/NATO_Vector_Graphics_%28NVG%29); it is standardized by NATO as STANAG 4733.
- The **FIPS 10-4** (Federal Information Processing Standards) standard, Countries, Dependencies, Areas of Special Sovereignty, and Their Principal Administrative Divisions, is a list of two-letter country codes used by the U.S. Government for geographical data processing in many publications, such as the CIA World Factbook. See https://en.wikipedia.org/wiki/List_of_FIPS_country_codes for a list of these codes.
- NATO Joint Military Symbology is the NATO standard for military map marking symbols. Originally published in 1986 as Allied Procedural Publication 6 (**APP-6B** and **APP-6C**). See http://armawiki.zumorc.de/files/NATO/APP-06(B)%20Joint%20Symbology.pdf for version B. See http://armawiki.zumorc.de/files/NATO/APP-6%28C%29.pdf for version C.
- **HTTP**: IETF Network Working Group “Hypertext Transfer Protocol – HTTP 1.1”, online at http://tools.ietf.org/html/rfc2616.
- **RFC3986**: IETF Network Working Group “Uniform Resource Identifier (URI): Generic Syntax”, online at https://www.ietf.org/rfc/rfc3986.

- Media type definition (**MIME**): IETF Network Working Group:
  - RFC 2045 - MIME formats and encodings
  - RFC 2046 - Definition of media types
  - RFC 2077 - Model top-level media type
  - RFC 7303 - XML Media Types
  - RFC 6657 - Update to MIME regarding "charset" Parameter Handling in Textual Media Types
  - RFC 6838 - Media type specifications and registration procedures
- **SSE**: Web Hypertext Application Technology Working Group “Server-sent events”, online at https://html.spec.whatwg.org/multipage/server-sent-events.html;
- **TS**: TypeScript, online at http://www.typescriptlang.org;
- **ECMAScript**: standardized by ECMA International in ECMA-262 and ISO/IEC 16262;
- **JSON**: standardized by ECMA International in ECMA-404.

3 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

Battle Space Object (BSO)

“A Battle Space Object (BSO) is a discrete entity; thing or being that does exist at a particular time or place on the battle space and has military or civilian significance.” (From NATO’s http://tide.act.nato.int/tidepedia/index.php?title=Battlespace_Object). See Track.

Datum

“A Geodetic system or geodetic datum is a coordinate system, and a set of reference points, used to locate places on the Earth (or similar objects).” (From http://en.wikipedia.org/wiki/Geodetic_datum).

ECEF

“ECEF ("Earth-Centred, Earth-Fixed"), also known as ECR ("Earth Centred Rotational"), is a Cartesian coordinate system, and is sometimes known as a "conventional terrestrial" system. It represents positions as an X, Y, and Z coordinate. The point (0, 0, 0) is defined as the center of mass of the Earth, hence the name Earth-Centred. Its axes are aligned with the International Reference Pole (IRP) and International Reference Meridian (IRM) that are fixed with respect to the surface of the Earth, hence the name Earth-Fixed. This term can cause confusion since the Earth does not rotate about the z-axis (unlike an inertial system such as ECI), and is therefore alternatively called ECR. The z-axis is pointing towards the north but it does not coincide exactly with the instantaneous Earth rotational axis. The slight "wobbling" of the rotational axis is known as polar motion. The x-axis intersects the sphere of the Earth at 0° latitude (Equator) and 0° longitude (Greenwich). This means that ECEF rotates with the earth and therefore, coordinates of a point fixed on the surface of the earth do not change. Conversion from a WGS84 Datum to ECEF can be used as an intermediate step in converting velocities to the North East Down coordinates system.” (From http://en.wikipedia.org/wiki/ECEF).

TACSIT

A Tactical Situation Display software component provides a display of relevant tactical information over and in conjunction with the geographic context of the information.

Track

A spatial object that is managed within the CMS, such as local radar contacts, radar contacts provided via external messages, tactical data points, waypoints, etc. See BSO.

WGS84

“The World Geodetic System (WGS) is a standard for use in cartography, geodesy, and navigation. It comprises a standard coordinate system for the Earth, a standard spheroidal reference surface (the datum or reference ellipsoid) for raw altitude data, and a gravitational equipotential surface (the geoid) that defines the nominal sea level. The latest revision is WGS 84 (aka WGS 1984, EPSG:4326), established in 1984 and last revised in 2004.” (From http://en.wikipedia.org/wiki/WGS84#A_new_World_Geodetic_System).

4 Symbols

None.

5 Additional Information

5.1 Problem statement (non-normative)

The domain of C2 Systems is characterized by a huge variety of underlying computing platforms, with different and often incompatible means of providing interactive displays. Standards-based services are essential for interoperable and open systems.

There is fairly broad agreement of what is considered the TACSIT software component of a tactical / strategic display system. The TACSIT component is the software that provides users awareness of entities in the operational space relative to a certain geospatial context. The TACSIT, by its nature, displays entities called tracks or BSO’s (Battle Space Object) in their proper geographic location overlaid on a visual representation of a map while including additional annotations and decision aides to support the operator. The TACSIT is distinct from other display applications that work around, or in conjunction with it.

The figure below provides an example of what the TACSIT software component is versus other typical tactical display applications and decision aids.

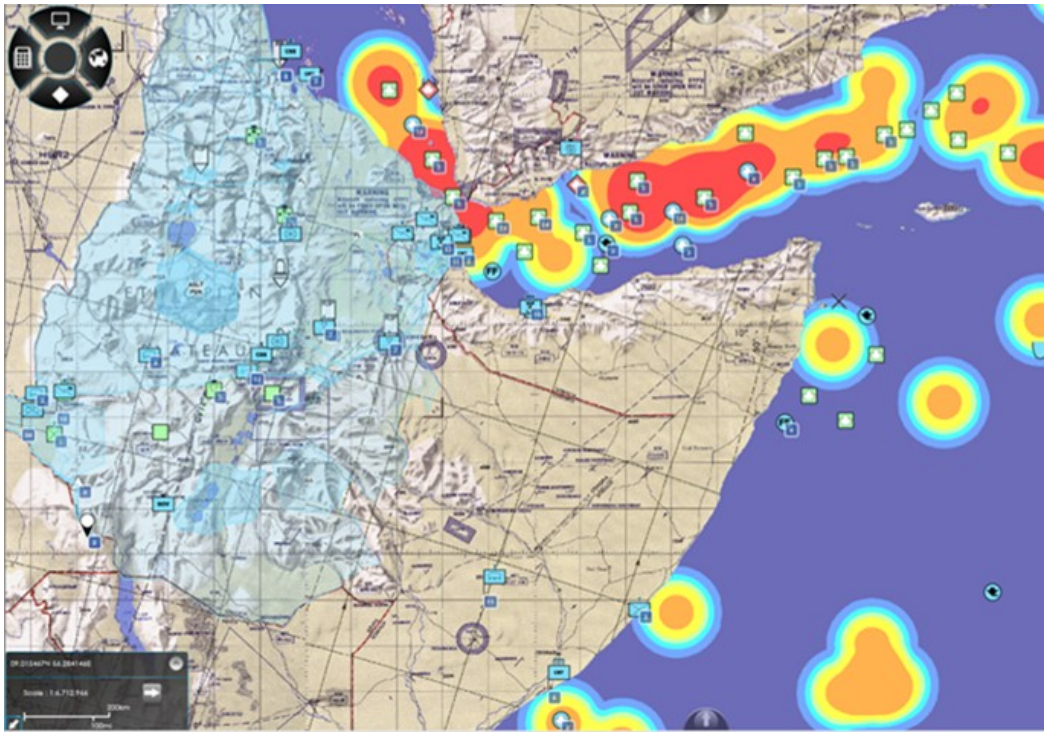


Figure 5.1 - Example of a TACSIT software component

There are many capabilities and services necessary to successfully implement a TACSIT software component. Within this broader context of TACSIT services, this specification is targeted at the Data Services necessary to exchange Tracks/BSO's data with TACSIT services.

More specifically, the considered data encompass all the data needed to draw Tracks/BSO's over a map as points (icons), multilines or surfaces.

The data necessary to infer the military icons, annotations and decision aides to be drawn must be exchangeable with TACSIT systems. Only the strictly required, limited-to-core data must be exchanged in order to avoid the transfer of unused data, e.g. between web browser and server and/or over poor communication means (satellite, radio…).

In the context of TEX, the needed data to draw a military symbol on a map (2D/3D) are:

- Data common to any Tracks/BSO: name, identifier, positions...; these data will be limited to the strict minimum;
- Data depending on the symbology used (APP-6B, 2525D, AIS...) they will be called "Categorization Data"; since such symbologies picture domain information, Categorization Data tend to hold business-specific information but the aim is to draw the symbol;
- Business-specific data that may be drawn/written around the symbol: they will considered as "Extended Data" and designed as key/value pairs.

Next, the grouping of Tracks/BSO's needs also to be considered. Indeed, operators of TACSIT may find it useful to group the Tracks/BSO's by Group name. These TACSIT groups provide a list of Tracks/BSO's, which is defined either by extension, i.e. by a set of items, or by intention, i.e. by a filter to be applied on all known Tracks/BSO's. Such TACSIT groups aid Situation Awareness, Common Operational Picture (COP), Recognized Ground/Air/Maritime Picture (RGP, RAP and RMP), orders and plans pictures. However, since this capability may be meaningless for some systems or harmful to performance, TACSIT groups are made optional in this specification.

5.2 Design rationale (non-normative)

5.2.1 Architecture Patterns

Even if this specification does not limit the use of its interfaces to any specific architecture, it has been designed with three specific patterns of architecture in mind: one with a standalone system and two based on distributed systems.

This section describes these three architecture patterns and how they relate to the other TACSIT standard, namely [TCI].

Note: the description of these patterns implies the concept of "channel" which encompasses any means of communication between components – e.g., intra-process calls, remote procedure invocations (CORBA, HTTP/REST, HTTP/SOAP...), publish/subscribe (DDS...) – and any necessary in-between items (ESB, Reverse Proxy...).

5.2.1.1 Standalone Architecture Pattern

In this first pattern, the application runs on one standalone computer. It has two types of components:

- The TACSIT component,
- The specific business applications such as business object forms, business objects lists and so forth.

The following figure depicts this pattern:

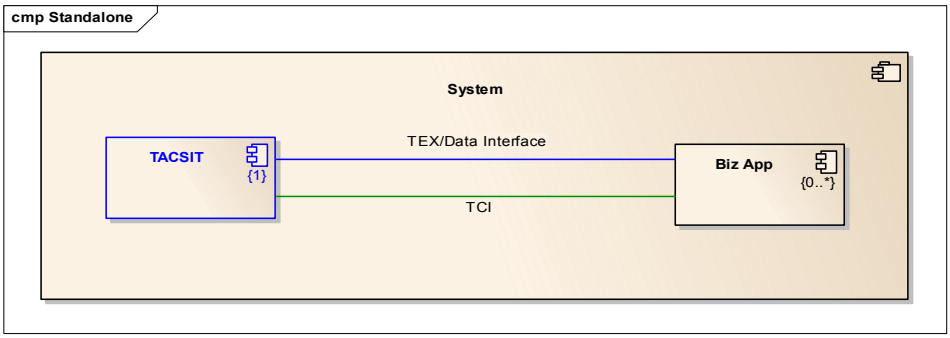


Figure 5.2 – Standalone Architecture Pattern (non-normative)

These two components communicate through two channels:

- The first conforms to [TCI] and allows a business application to control the display of the TACSIT component as well as to get the entity selection currently set by the user;
- The second one conforms to the “Data Interface” package introduced in the present specification and allows the TACSIT component and the business application components to exchange displayed data and/or get callback on the modification thereof.

Both channels work together based on the common Entity concept (which designs a Track/BSO).

5.2.1.2 Distributed Without TACSIT Back-end Architecture Pattern

In this second pattern, the application is made up of graphical user interfaces (GUI) and one business back-end. It is constituted of the following components:

- The TACSIT component,
- The specific business HMIs such as business object forms, business objects lists and so forth,
- The business back-end or business server that serves business data to the business HMIs and display data to the TACSIT component.

The following figure depicts this pattern:

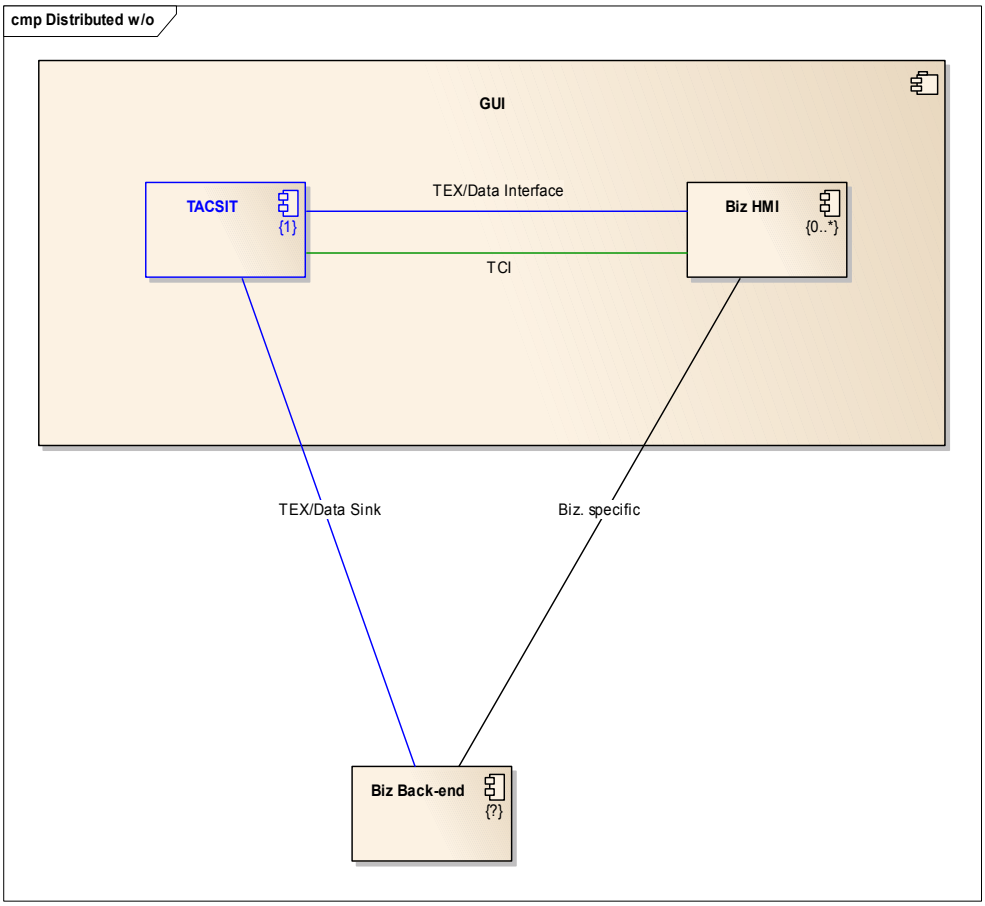


Figure 5.3 – Distributed Without TACSIT back-end Architecture Pattern (non-normative)

These components communicate through the following channels:

- A first channel conforms to [TCI] and allows a business HMI to control the display of the TACSIT component as well as to get the entity selection currently set by the user;
- A second channel conforms to the “Data Interface” package introduced in the present specification and allows the TACSIT component and the HMI components to exchange displayed data and/or get callback on the modification thereof;
- A third channel takes place between the business HMI and the business back-end and is considered as proprietary in the solution;
- A last channel allows the TACSIT component to fetch to-be-displayed data from the business back-end; it conforms to the “Data Sink” package introduced in the present specification.

TCI's and TEX' channels work jointly based on the common Entity concept.

5.2.1.3 Distributed With TACSIT Back-end Architecture Pattern

In this third pattern, the application is made up of graphical user interfaces (GUI), one business back-end and one TACSIT back-end. With regards to the preceding pattern, the back-end is here split in two back-ends: one for the TACSIT display data and the other one for the business data. The pattern is thus constituted of the following components:

- The TACSIT component,
- The specific business HMIs such as business object forms, business objects lists and so forth,
- The business back-end or business server that serves business data to the business HMIs,

- The TACSIT back-end or TACSIT server that serves display data to the TACSIT component.

The following figure depicts this pattern:

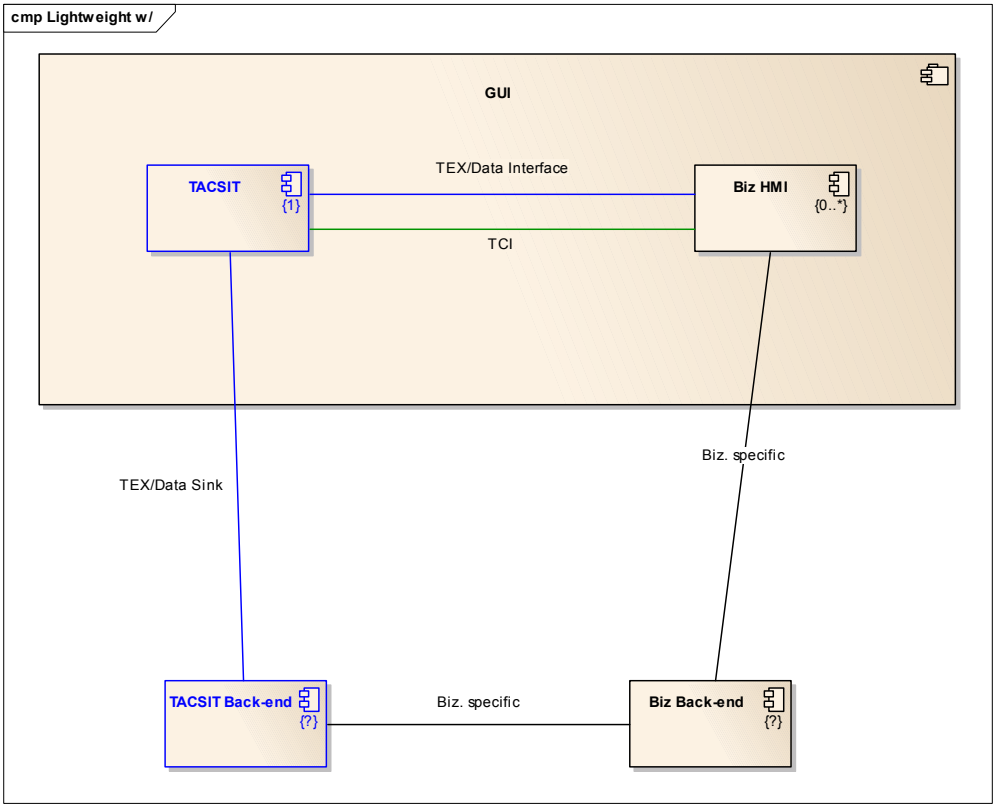


Figure 5.4 – Distributed With TACSIT back-end Architecture Pattern (non-normative)

These components communicate through the following channels:

- A first channel conforms to [TCI] and allows a business HMI to control the display of the TACSIT component as well as to get the entity selection currently set by the user;
- A second channel conforms to the “Data Interface” introduced in the present specification and allows the TACSIT component and the HMI components to exchange displayed data and/or get callback on the modification thereof;
- A third channel takes place between the business HMI and the business back-end and is considered as proprietary in the solution;
- A fourth channel allows the TACSIT component to fetch to-be-displayed data from the TACSIT back-end; it conforms to the “Data Sink” interfaces introduced in the present specification;
- A last (optional) channel allows both back-ends to exchange information preventing them to serve inconsistent data; this channel is out of the scope of TEX.

TCI's and TEX' channels work jointly based on the common Entity concept.

5.2.2 Payload vs. Interface vs. Data Sink

As described in the above section, TEX specifies interfaces to push and pull data to and from a TACSIT system.

These interfaces are either implemented by a TACSIT system and used by an external system, or used by a TACSIT system and implemented by an external system.

In the first case (implemented by a TACSIT system), the interfaces allow external systems to push and pull data to and from a TACSIT system.

In the second case (implemented by an external system), the interfaces allow a TACSIT system to get data from the external system; in this case, the external system is named a “data sink” for the TACSIT system and the matching interface “data sink interface” while the preceding interface are simply named “data interfaces”.

An important point here is that TACSIT data interfaces and data sink interface share the same specification of the transported data (the payload).

TEX interfaces are therefore designed by splitting the data (payload) from the interfaces and by splitting data interfaces from data sink interfaces. In that way, the transported data, i.e. the payload in the interfaces and data sink, are designed in a specific package (namely DataPayload) while the interfaces are designed in another package (namely DataInterface) and the interfaces for the data sink in a third package (namely DataSink).

This design allows the use of different PSMs for payload, data interfaces and data sink interfaces.

The following figure show a system based on what has already been introduced in the preceding section as a Lightweight Without TACSIT Back-end Architecture Pattern. The TACSIT system provides data interfaces to other browser-based applications (Biz GUI in the figure) and requests data sink interface implemented at server level (Biz Server in the figure). This last interface is designed to get data from the field and needs to perform fast enough to ensure a correct understanding of the tactical situation.

In this example, the payload PSM could be XML, the interface PSM TypeScript and the data sink PSM HTTP.

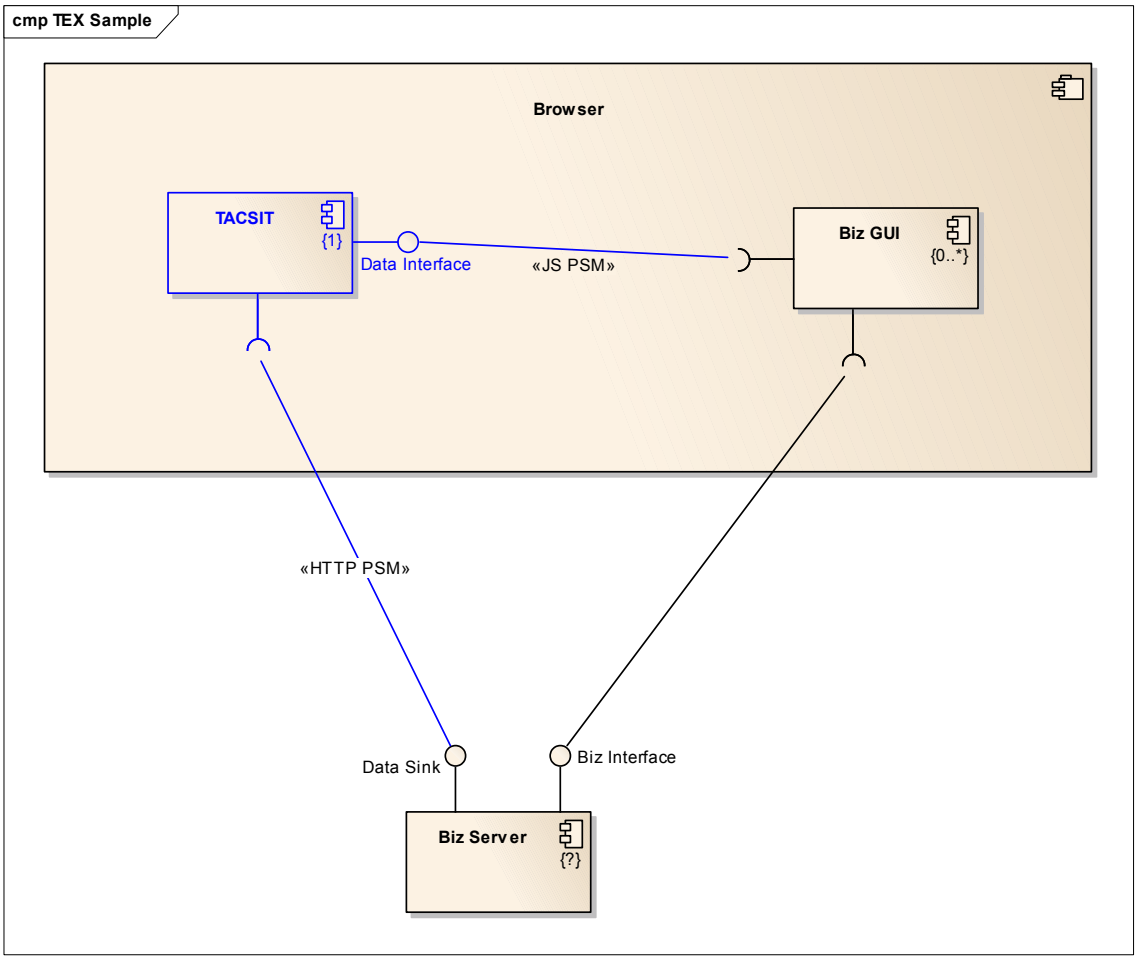


Figure 5.5 – Example of use of a web-based TACSIT system (non-normative)

5.2.3 One Namespace to Rule Them All

This specification adds new capabilities to [TCI] by providing new packages aside of the previous TCI ones. As depicted in the following figure, TEX does not add a new namespace but enhance *org.omg.tacsit* with three new

packages: DataPayload, DataInterface and DataSink as introduced in the preceding section. The TCI and TEX level exists so only for the sake of documentation management as depicted in the following diagram by dashed boxes.

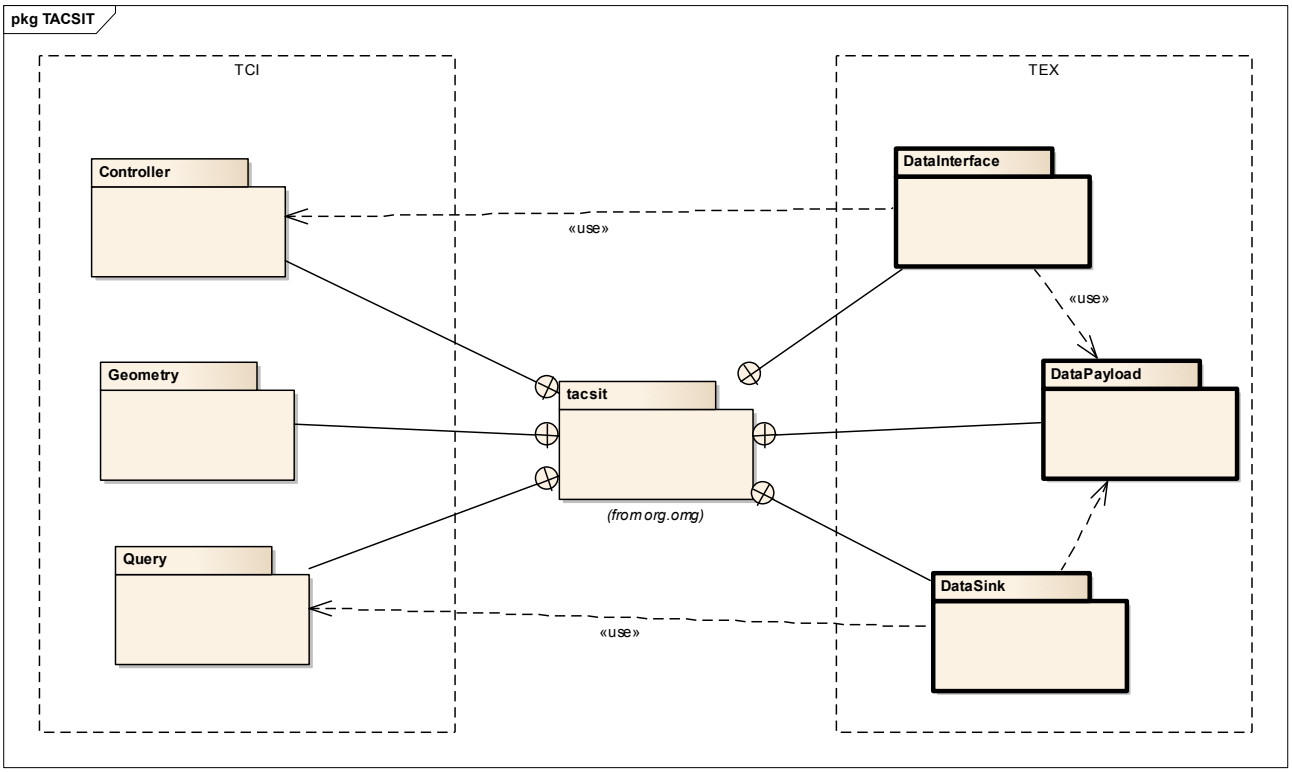


Figure 5.6 – Enhancement to TACSIT TCI by TACSIT TEX (non-normative)

5.2.4 Dependencies

The DataInterface package uses the DataPayload package. Incidentally, it also uses the Controller package (more specifically, it subclasses the Entity interface) defined by the [TCI] specification.

The DataSink package also uses the DataPayload package yet uses the Query package from TCI (more specifically the EntityQuery interface) from TCI.

The DataPayload PIM is described in section 7, the DataInterface PIM in section 8 and the DataSink PIM in section 9.

5.2.5 Optional Capabilities

Some capabilities have been considered as optional, meaning that conforming implementations are not obliged to implement them. Yet, in these cases, the matching method are designed to be implemented and to return the NotImplemented exception. A user of such a method needs to be aware that the targeted TACSIT implementation(s) may or may not offer these capabilities.

The optional capabilities are:

- The entity grouping capability may be restricted because it may be expensive to implement and/or costly in term of memory and CPU consumption and/or not useful for the specific domain of interest; this restriction is done either by not allowing the creation of new groups or by limiting the number of such groups; whatever the way the restriction is done, a TACSIT system has always at least one such group to hold entities;
- A TACSIT system may be unable to accept entity modification from an external system (for instance, because in some system it may be considered that it is not the TACSIT system's role to maintain data coherence of entities across a system with multiple TACSIT instances): all interfaces that change entities or groups can thus return the NotImplemented exception;
- Listeners may offer a rate at which the client is informed of the changes.

5.2.6 Extensibility

This standard holds an extensible capability. Here, 'extensible' means that this capability may be extended in a non-foreseen way while keeping the conformance of the implementation with regards to the standard.

This extensible capability is the specification of the data of an entity used to choose the symbol used to draw the given entity. The extensibility is provided through:

- An abstract class, CategorizationData (See Clause 7), that is to be sub-classed for each new categorization;
- A couple of tags, SymbolSet and SymbolId, that may be used by the Data Payload PSMs to generically implement such categorization: SymbolSet is a unique string that identify the specified symbology (e.g. “app6a” for the APP-6A symbology), and SymbolId is a human-readable specification of the method to work out a symbol id from the attributes.

A typical use of these tags is to specify the way a categorization data in transported as a string built from the SymbolSet, the character “:” and the string resulting of the application of SymbolId to the attributes of the actual CategorizationData class.

A CategorizationData for which a SymbolSet is already known in this standard shall not be defined otherwise by an implementation.

5.2.7 Use patterns

This section exemplifies potential patterns on how to use the interfaces specified in this document with sequence diagrams. These use patterns are not normative, meaning that the interfaces may be used otherwise. Specifically, they are designed in the context of the “Distributed Without TACSIT back-end” Architecture Pattern (see Section 5.2.1.2). Nevertheless, they may easily be translated to any of the other Architecture Patterns in Section 5.2.1 and most probably to any other pattern.

In the following diagrams, colors are used to point out items related to the TEX standard (blue-colored) and items related to the TCI standards (green-colored). Nevertheless, these colors are meant to ease the reading of these diagrams and do not bring any new semantics to them.

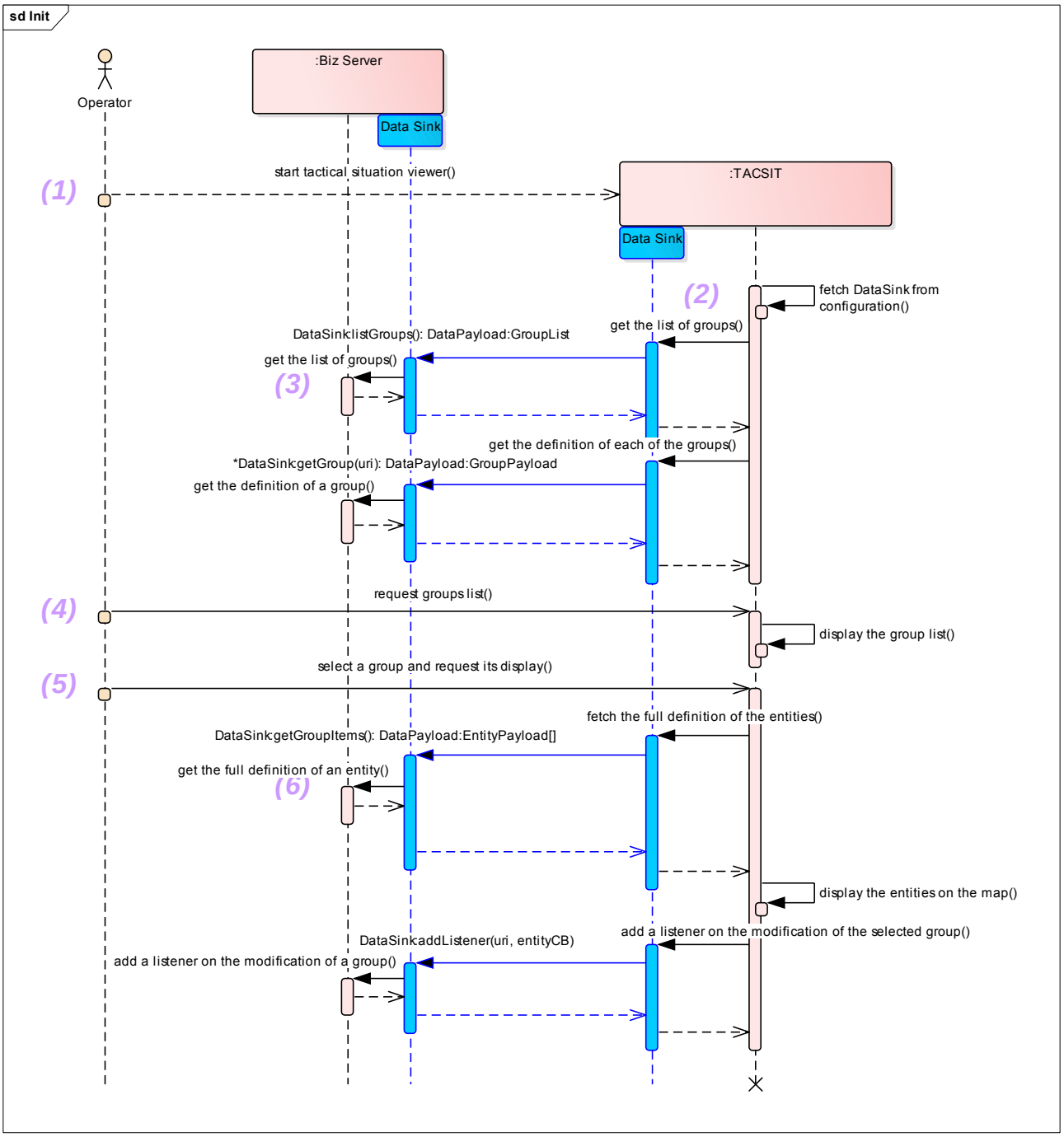
5.2.7.1 Initialization

In this example of initialization, an operator launches a TACSIT situation viewer, requests the list of groups (overlays) and ask for the display of one of them.

More specifically:

1. The operator starts the TACSIT system, which consequently is created;
2. The TACSIT system retrieves from its configuration the connection means (depending on the PSM used) of the DataSink interface of the Business server;
3. The TACSIT system invokes the DataSink interface:
   - To get the list of groups, and
   - To iteratively get the definition of each of these groups (meta data as well as list of entity pointers);
4. The operator requests the listing of the groups; The TACSIT system uses the preceding groups definition to display an HMI listing the groups name, permissions, entities number and so forth;
5. The operator selects a group and requests its display on the map;
6. The TACSIT system:
   - Invokes the DataSink interface to fetch the full definition of the entities of the selected group;
   - Displays these entities on the map using the data provided by the DataPayload PIM; and
   - Invokes the DataSink interface to add a listener on the modification of the selected group in order to refresh the entities on the map.

[TEX-1 – Figures 5-7 to 5-11 have some strange notation]


sd Init
Operator
:Biz Server
Data Sink
:TACSIT
Data Sink
(1)
start tactical situation viewer()
(2)
fetch DataSink from configuration()
get the list of groups()
DataSink:listGroups(): DataPayload:GroupList
get the list of groups()
(3)
get the definition of each of the groups()
*DataSink:getGroup(uri): DataPayload:GroupPayload
get the definition of a group()
(4)
request groups list()
display the group list()
(5)
select a group and request its display()
fetch the full definition of the entities()
DataSink:getGroupItems(): DataPayload:EntityPayload[]
get the full definition of an entity()
(6)
display the entities on the map()
add a listener on the modification of the selected group()
DataSink:addListener(uri, entityCB)
add a listener on the modification of a group()

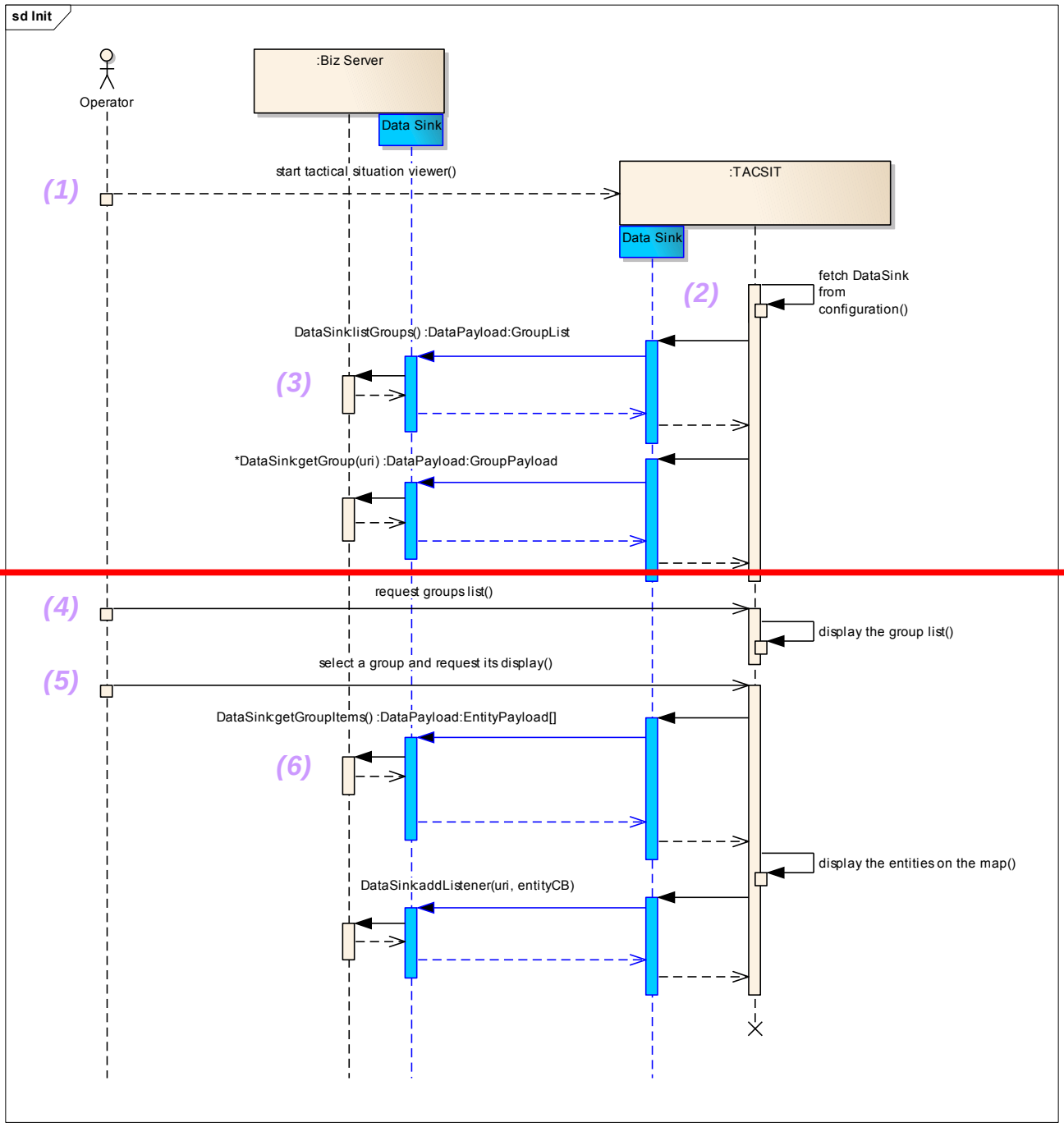


Figure 5.7 – The 'Initialization' Use pattern (non-normative)

5.2.7.2 Update of a Single Track/BSO

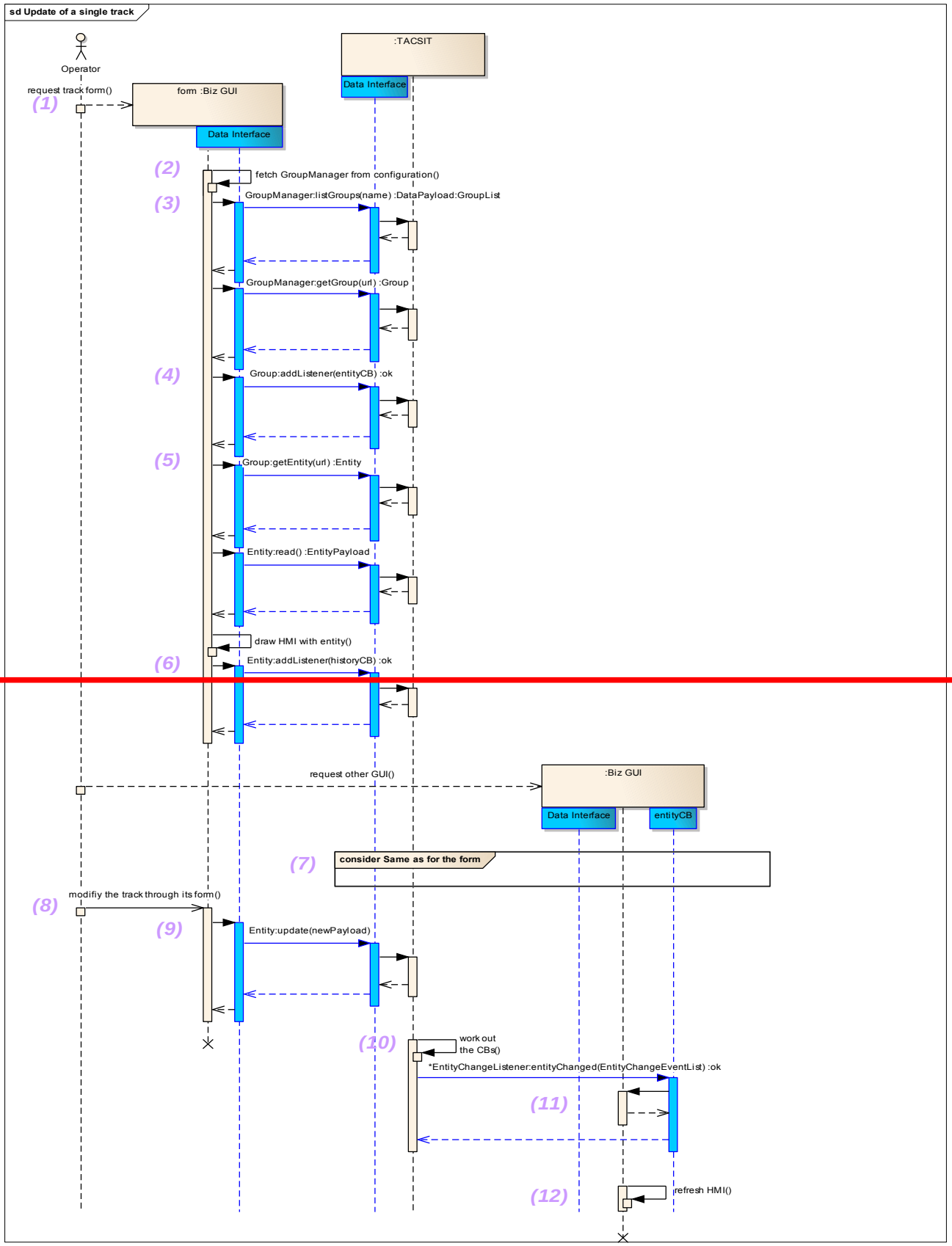
This use pattern is the follow up of the preceding one. The operator launches two business GUIs displaying the same entity, uses one of them to modify this entity. The second GUI receives a notification of this modification and refreshes itself.

More specifically:

1. The operator starts her/his entity form (called “Biz GUI”) providing the group name and the entity named s/he wants to work on;
2. The entity form retrieves from its configuration the connection means (depending on the PSM used) of the DataInterface:DataManager interface of the TACSIT system (this component was fully initialized during the preceding use pattern);

3. The entity form uses this DataInterface interface to get the list of groups and all the data for the specific group to be used (from its name);
4. The entity form uses this DataInterface interface to add a listener on the modifications of this group;
5. The entity form uses this DataInterface interface to get the complete definition of the wanted entity from its name and fill in the form with them;
6. The entity form uses this DataInterface interface to add a listener on the modifications of this entity;
7. The operator starts another GUI displaying the same entity (e.g., a grid): follows then the same kind of sequence as from (2) to (6);
8. S/he updates the entity with her/his form and saves these modifications;
9. The form uses its DataInterface interface to update the entity in the TACSIT system;
10. The TACSIT component works out the callbacks that need to be fired;
11. and iteratively invokes them, including the other GUI's callback interface provided to the business server at form initialization (see bullet 6);
12. Eventually, the other GUI refreshes itself with the new data provided with the callback.

[TEX-1 – Figures 5-7 to 5-11 have some strange notation]


sd Update of a single track
Operator
:TACSIT
Data Interface
request track form()
(1)
form :Biz GUI
Data Interface
(2)
fetch GroupManager from configuration()
(3)
GroupManager:listGroups(name) :DataPayload:GroupList
GroupManager:getGroup(url) :Group
(4)
Group:addListener(entityCB) :ok
(5)
Group:getEntity(url) :Entity
Entity:read() :EntityPayload
draw HMI with entity()
(6)
Entity:addListener(historyCB) :ok
request other GUI()
:Biz GUI
Data Interface
entityCB
(7)
consider Same as for the form
modifiy the track through its form()
(8)
(9)
Entity:update(newPayload)
(10)
work out
the CBs()
*EntityChangeListener:entityChanged(EntityChangeEventList) :ok
(11)
(12)
refresh HMI()

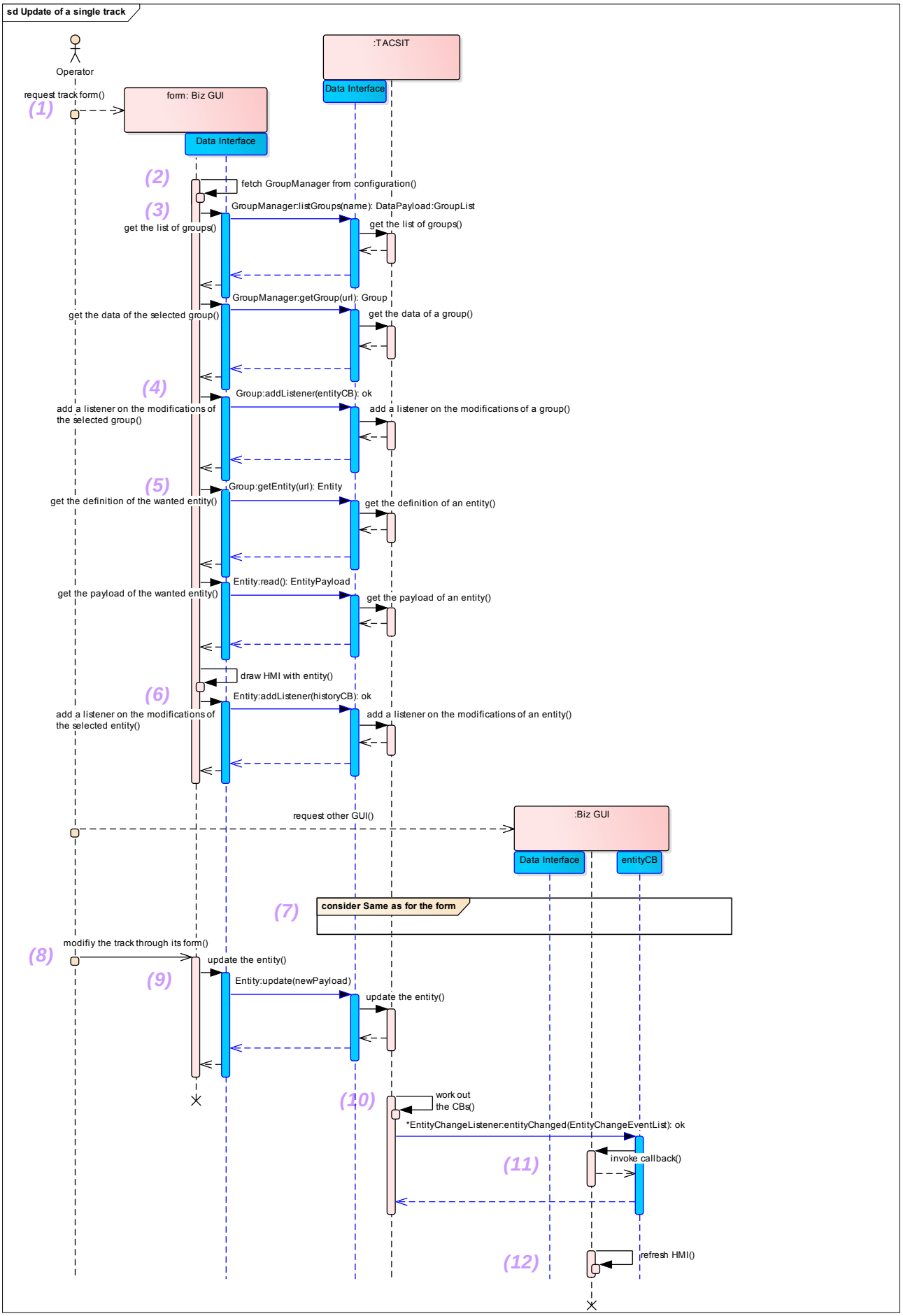

sd Update of a single track
Operator
:TACSIT
Data Interface
request track form()
(1)
form: Biz GUI
Data Interface
(2)
fetch GroupManager from configuration()
(3)
GroupManager:listGroups(name): DataPayload:GroupList
get the list of groups()
get the list of groups()
GroupManager:getGroup(url): Group
get the data of the selected group()
get the data of a group()
(4)
Group:addListener(entityCB): ok
add a listener on the modifications of the selected group()
add a listener on the modifications of a group()
(5)
Group:getEntity(url): Entity
get the definition of the wanted entity()
get the definition of an entity()
Entity:read(): EntityPayload
get the payload of the wanted entity()
get the payload of an entity()
draw HMI with entity()
(6)
Entity:addListener(historyCB): ok
add a listener on the modifications of the selected entity()
add a listener on the modifications of an entity()
request other GUI()
:Biz GUI
Data Interface
entityCB
(7)
consider Same as for the form
(8)
modifiy the track through its form()
update the entity()
(9)
Entity:update(newPayload)
update the entity()
(10)
work out the CBs()
*EntityChangeListener:entityChanged(EntityChangeEventList): ok
(11)
invoke callback()
(12)
refresh HMI()

Figure 5.8 – The 'Update of a Single Track/BSO' Use pattern (non-normative)

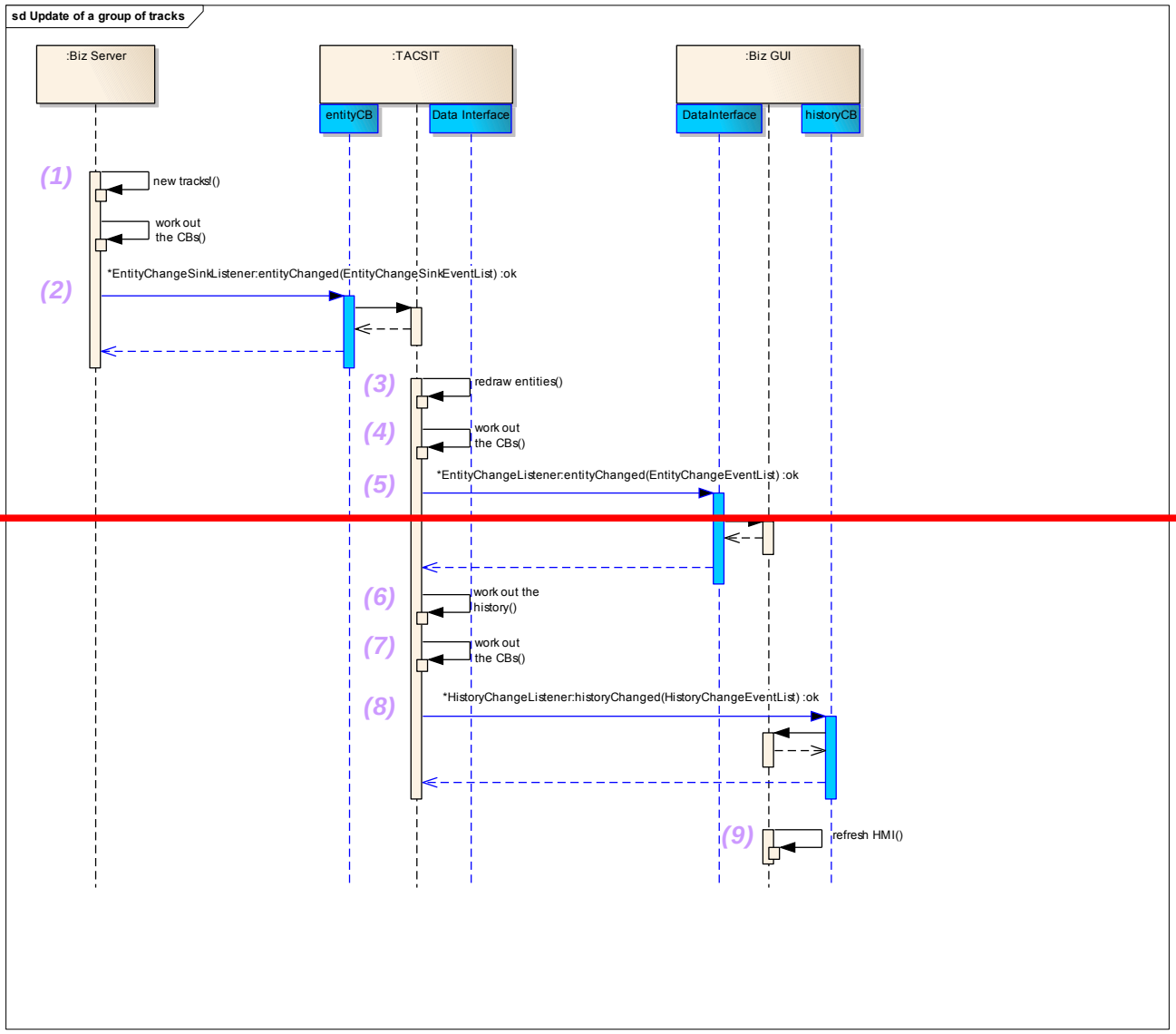
5.2.7.3 Update of a Group of Tracks/BSO's

This use pattern is the follow up of the two preceding ones. The business server receives some updates either from outside of the system (e.g., from a tactical data link) or from inside of the system (e.g., another operator). It sends these updates to the TACSIT components that in turn send updates to the business GUIs for purposes of refreshing.

More specifically:

1. The business server receives new tracks and works out the list of callbacks that need to be invoked to inform the DataSink clients;
2. The business server iteratively invokes these callbacks through the DataSink interface (set up during the initialization of the TACSIT system);
3. The TACSIT system redraws the updated entities on the map;
4. The TACSIT system works out the list of entity callbacks that needs to be invoked;
5. The TACSIT system iteratively invokes these callbacks through the DataInterface interface (set up during the initialization of the business GUIs);
6. The TACSIT system works out the new history for the updated entities;
7. The TACSIT system works out the list of history callbacks that needs to be invoked;
8. The TACSIT system iteratively invokes these callbacks through the DataInterface interface (set up during the initialization of the business GUIs);
9. The Business GUIs refresh themselves from the data provided along with the callbacks.

[TEX-1 – Figures 5-7 to 5-11 have some strange notation]


sd Update of a group of tracks
:Biz Server
:TACSIT
entityCB
Data Interface
:Biz GUI
DataInterface
historyCB
(1)
new tracks!()
work out the CBs()
(2)
*EntityChangeSinkListener:entityChanged(EntityChangeSinkEventList) :ok
(3)
redraw entities()
(4)
work out the CBs()
(5)
*EntityChangeListener:entityChanged(EntityChangeEventList) :ok
(6)
work out the history()
(7)
work out the CBs()
(8)
*HistoryChangeListener:historyChanged(HistoryChangeEventList) :ok
(9)
refresh HMI()

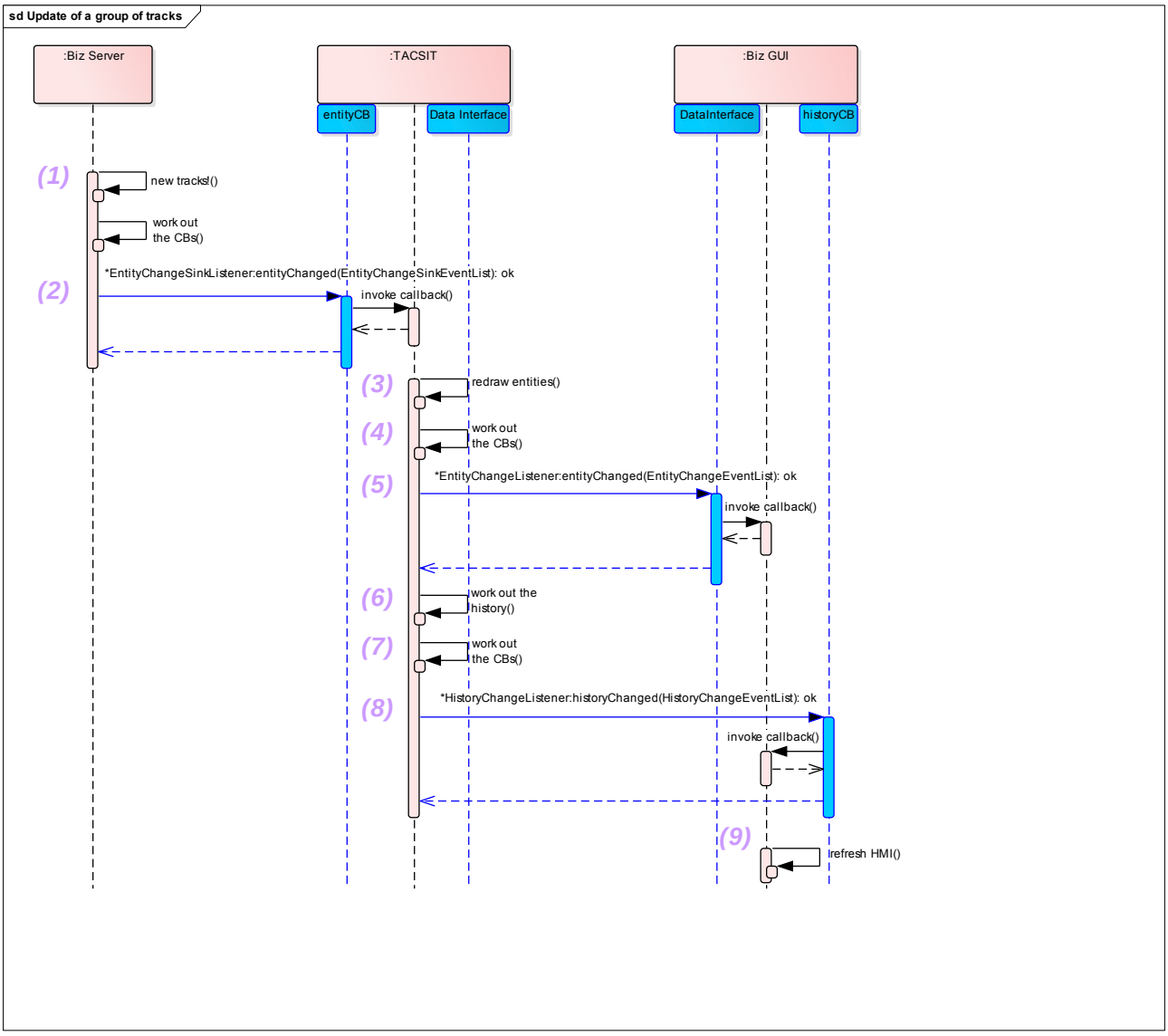


Figure 5.9 – The 'Update of a Group of Tracks/BSOs' Use pattern (non-normative)

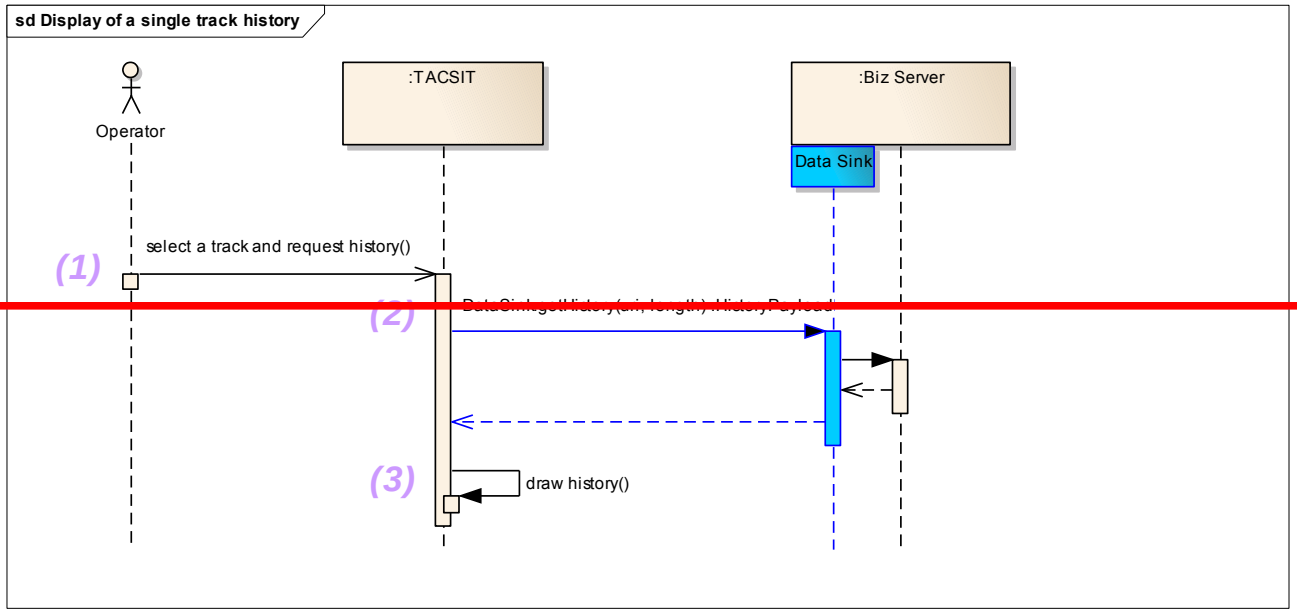
5.2.7.4 Display of a Single Track/BSO History

This use pattern is the follow up of the first pattern. The operator requests the display of the history path of an entity.

More specifically:

1. The operator asks the TACSIT system for a display of the history path of an entity (however this is done through the GUI);
2. The TACSIT system invokes the business server for this information through its DataSink interface;
3. The TACSIT system draws the history path of the entity.

[TEX-1 – Figures 5-7 to 5-11 have some strange notation]



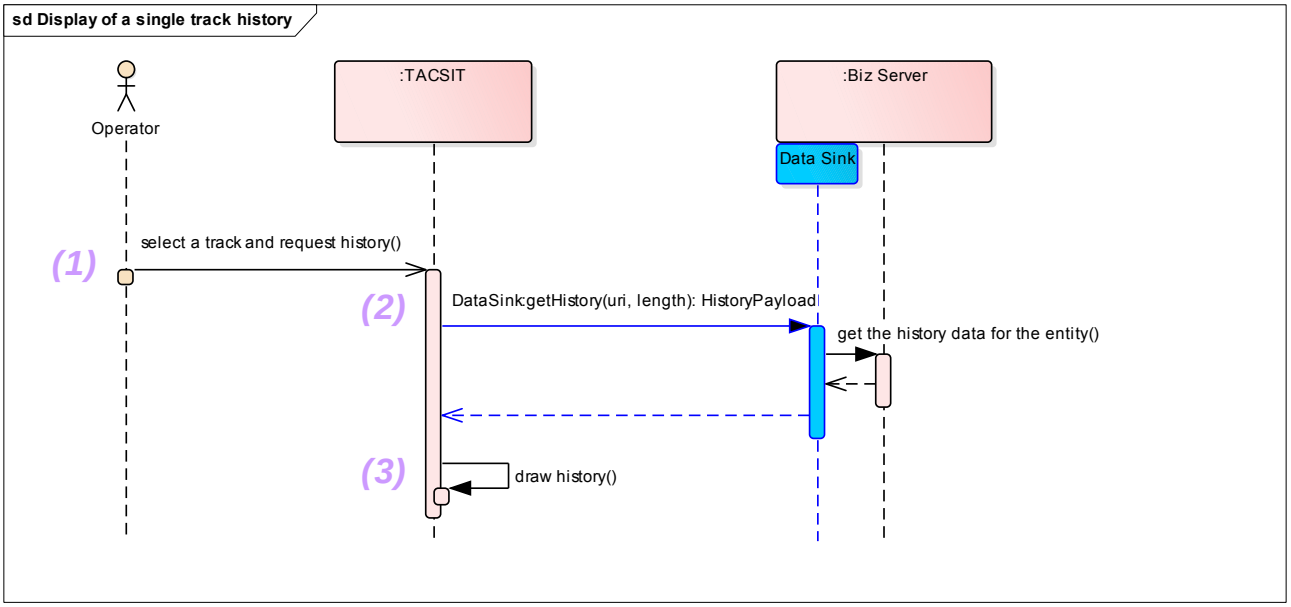


Figure 5.10 – The 'Display of a Single Track/BSO History' Use pattern (non-normative)

5.2.7.5 TEX and TCI

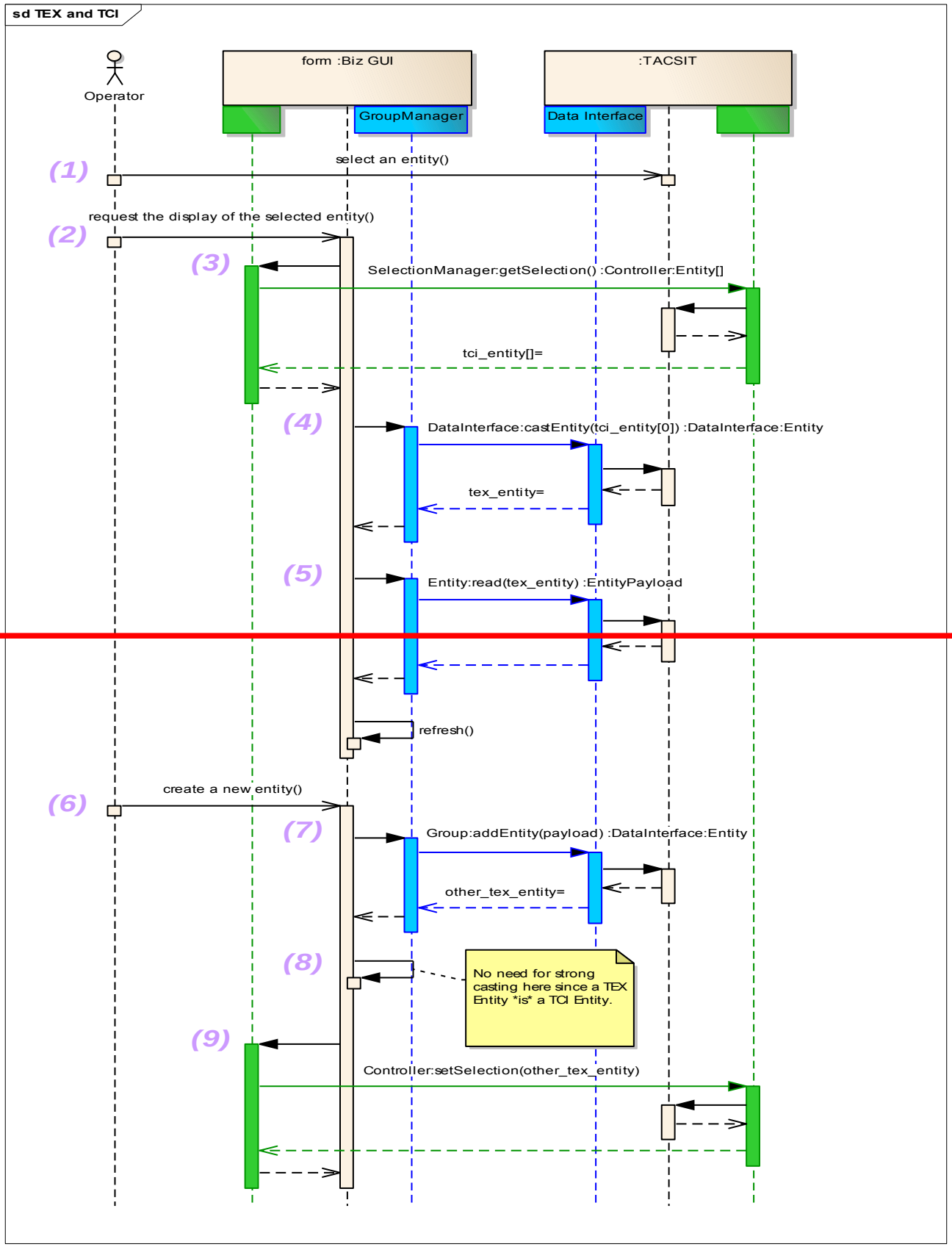
This use pattern is the follow up of the first pattern. The operator plays around with its entity form and the TACSIT selection to show off. The main point here is to illustrate the interoperability between TEX and TCI.

More specifically:

1. The operator selects an entity in the TACSIT component;
2. The operator asks the entity form to display the currently selected entity (by whatever means this is done through the GUI);
3. The form invokes the TACSIT system to get the selected entities through the Controller interface (TCI);
4. The form invokes the TACSIT system to translate (“cast”) the preceding TCI entity onto a TEX one;
5. The form invokes the TACSIT system to get the entity payload of the obtained TEX entity and refreshes itself;
6. Now, the operator requests the form to create a new entity and modify it;

7. The form invokes the TACSIT system to create a new entity with the new payload; the TACSIT component outputs the created entity;
8. The form does not need to translate the new TEX Entity onto a TCI Entity since TEX Entity is a subclass of TCI Entity;
9. The form may now invoke the TACSIT system to set the selection to the new entity through the Controller interface (TCI).

[TEX-1 – Figures 5-7 to 5-11 have some strange notation]


sd TEX and TCI
Operator
form :Biz GUI
GroupManager
:TACSIT
Data Interface
(1)
select an entity()
(2)
request the display of the selected entity()
(3)
SelectionManager:getSelection() :Controller:Entity[]
tci_entity[]=
(4)
DataInterface:castEntity(tci_entity[0]) :DataInterface:Entity
tex_entity=
(5)
Entity:read(tex_entity) :EntityPayload
refresh()
(6)
create a new entity()
(7)
Group:addEntity(payload) :DataInterface:Entity
other_tex_entity=
(8)
No need for strong casting here since a TEX Entity *is* a TCI Entity.
(9)
Controller:setSelection(other_tex_entity)

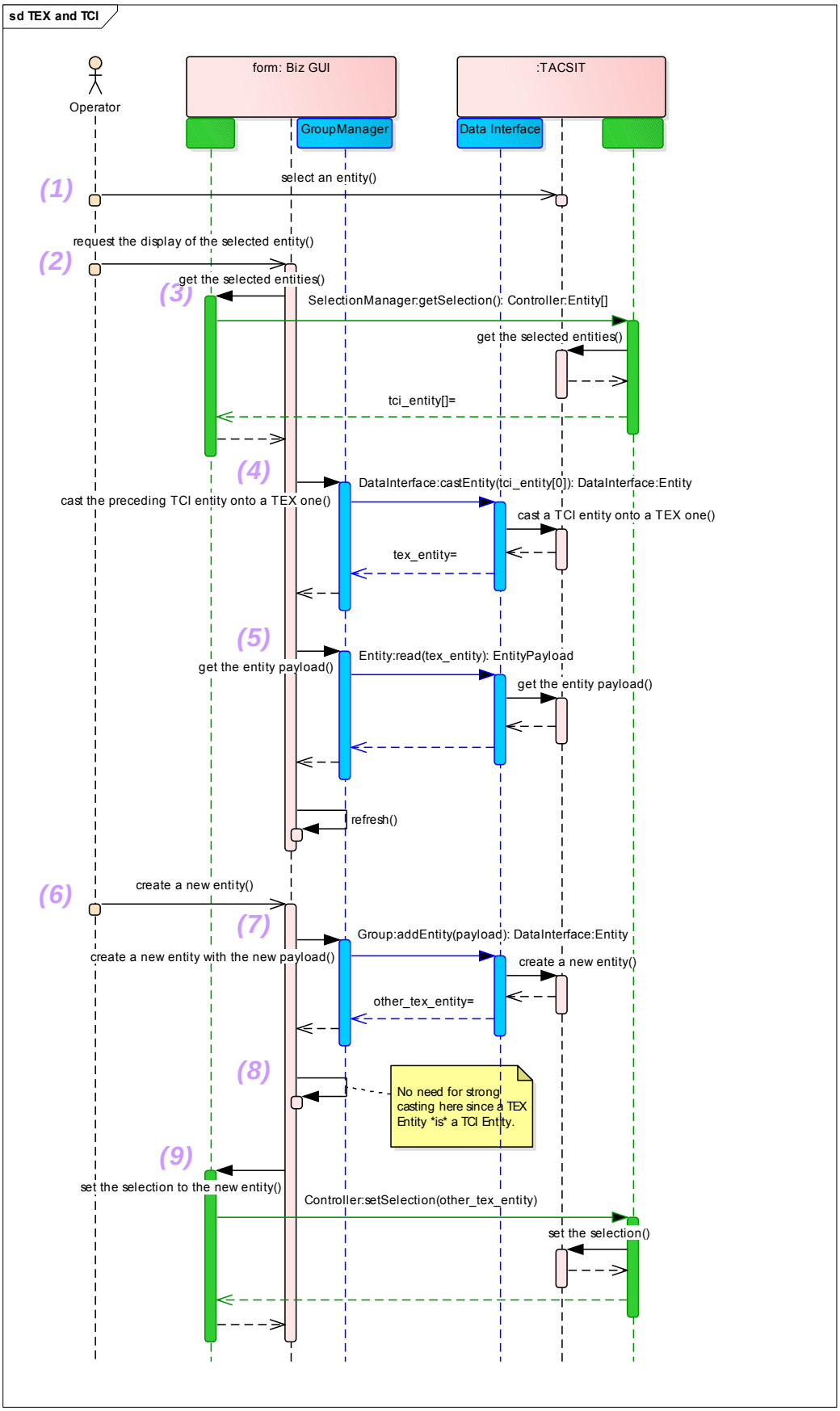


Figure 5.11 – The 'TEX and TCI' Use pattern (non-normative)

5.2.8 On Giant's Shoulders

This specification reuses several structures from [OARIS]. In order to avoid hard-to-maintain dependencies between the two set of standards, these structures have been copied rather than referenced.

Nevertheless, the specification still does maintain documentary references to OARIS as notes in diagram and text in order to keep the design rationale behind the structures.

As for the overlap with TCI, it is done by references due to the closeness of the two standards.

5.3 Changes to Adopted OMG Specifications

This specification add the following PSM to the TCI specification:

- DDS PSM,
- C# PSM,
- TypeScript PSM.

These PSM are presented in Annex B.

5.4 Acknowledgments

The following companies submitted this specification:

- Thales,
- SimVentions Inc,
- BAE Systems.

The following companies supported this specification:

- General Dynamics.

This page intentionally left blank.

6 Data Payload Platform-Independent Model

The DataPayload package defines the data which may be exchanged to/from a TACSIT system.

It is subdivided into 6 other packages:

- EntityPayload for the entities;

- CategorizationData for the data of the entity which are used to draw the entities;

- GroupPayload for the groups of entities;

- EntityHistory for the timed status of the entities;

- CallbackData contains the classes which are exchanged in callback methods, whatever the direction of the interfaces; and

- A utility package containing the general purpose data types and enumerations.

-

Note: Implementation details (such as getter and setter if any) are postponed to PSM-level choices.

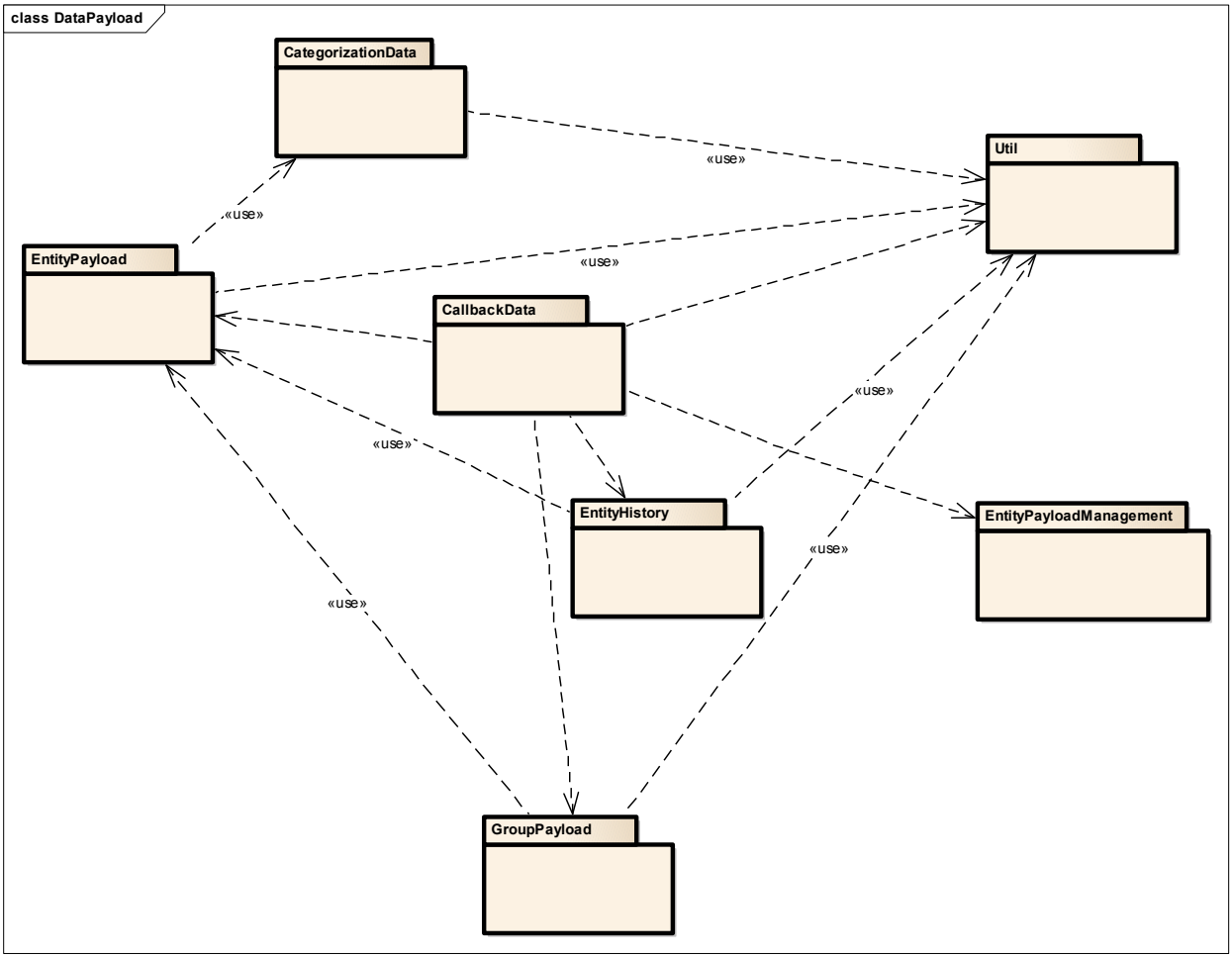


Figure 6.1 - DataPayload (Class diagram)

6.1 Util

This package contains several datatypes and enumeration used by other packages.

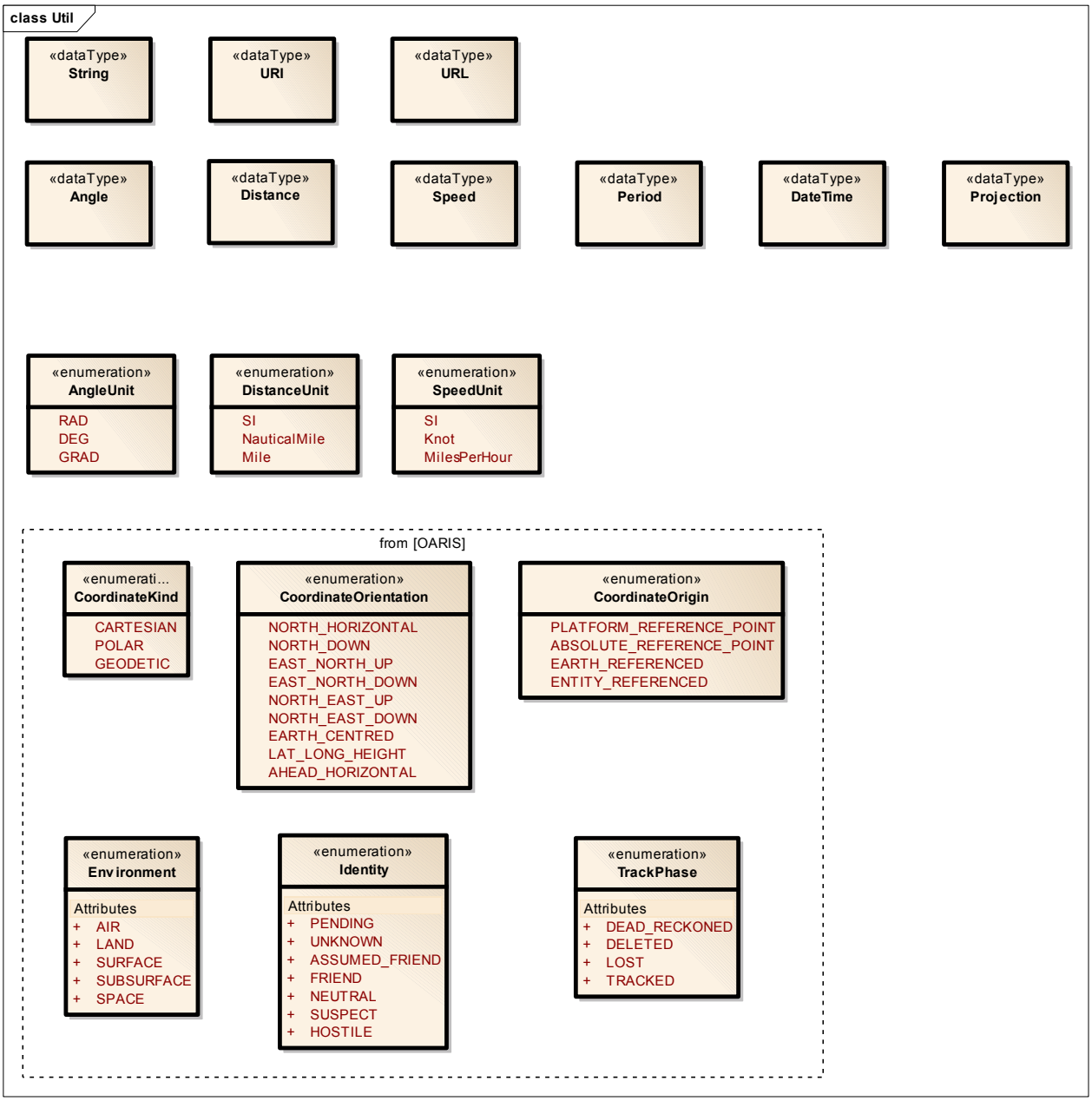


Figure 6.2 - Util (Class diagram)

6.1.1 AngleUnit (Enumeration)

Enumeration of the known Angle Units: radian, degree and grad.

Connections

Connector	Source	Target	Notes
Aggregation angleUnit Source -> Destination	AngleUnit	CoordinateUnits	The angle unit used by the group. RAD is the default.

Attributes

Attribute	Notes	Default
RAD	Radian.	
DEG	Degree.	
GRAD	Grad.	

6.1.2 CoordinateKind (Enumeration)

Enumeration of the coordinate systems, which compliant implementations may implement.

(from [OARIS])

Connections

Connector	Source	Target	Notes
Aggregation coordKind Source -> Destination	CoordinateKind	CoordinateUnits	The coordinate kind unit used by the group. GEODETIC is the default.

Attributes

Attribute	Notes	Default
CARTESIAN	Cartesian Coordinates (x, y, z).	
POLAR	Polar coordinates (azimuth, elevation, range).	
GEODETIC	Geodetic coordinates (latitude, longitude, altitude).	

6.1.3 CoordinateOrientation (Enumeration)

Enumeration of the coordinate systems, which compliant implementations may use. A compliant implementation may not fully support all of these coordinate systems.

(from [OARIS])

Connections

Connector	Source	Target	Notes
Aggregation coordOrientation Source -> Destination	CoordinateOrientation	CoordinateUnits	The coordinate orientation used by the group. Default depend upon the coordinate kind.

Attributes

Attribute	Notes	Default
NORTH_HORIZONTAL	Valid for Polar Coordinate Kind Azimuth has origin (0.0) at North, positive clockwise, measured in the horizontal plane Elevation has origin (0.0) at the Horizontal, positive up, measured in the vertical plane.	
NORTH_DOWN	Valid for Polar Coordinate Kind Azimuth has origin (0.0) at North, clockwise positive, measured in the horizontal plane Elevation has origin (0.0) when pointing directly down, and 180.0 degrees when pointing directly up, measured in the vertical plane.	
EAST_NORTH_UP	Valid for Cartesian coordinate type x is positive to the East y is positive to the North z is positive up.	
EAST_NORTH_DOWN	Valid for Cartesian coordinate type x is positive to the East y is positive to the North z is positive down.	
NORTH_EAST_UP	Valid for Cartesian coordinate type x is positive to the North y is positive to the East z is positive up.	
NORTH_EAST_DOWN	Valid for Cartesian coordinate type x is positive to the North y is positive to the East z is positive down.	
EARTH_CENTRED	Cartesian system with origin at centre of the Earth (absolute reference point) x positive through Greenwich meridian y positive through 90 degrees east (of Greenwich meridian) z positive through north pole x & y are in the equatorial plane.	
LAT_LONG_HEIGHT	WGS84 has unique well-defined orientation (NIMA Technical Report TR8350.2).	
AHEAD_HORIZONTAL	Valid for Polar Coordinate Kind Azimuth has origin (0.0) ahead of the relevant entity, positive clockwise, measured in the horizontal plane Elevation has origin (0.0) at the Horizontal, positive up, measured in the vertical plane.	

6.1.4 CoordinateOrigin (Enumeration)

Enumeration of the origins of the coordinate system.

(from [OARIS])

Connections

Connector	Source	Target	Notes
Aggregation coordOrigin Source -> Destination	CoordinateOrigin	CoordinateUnits	The coordinate Origin used by the group. ABSOLUTE_REFERENCE_POI

			NT is the default.

Attributes

Attribute	Notes	Default
PLATFORM_REFERENCE_POINT	The origin of the coordinate system is ‘well known’ reference point for the platform (on which the TACSIT system reside).	
ABSOLUTE_REFERENCE_POINT	The origin for the coordinate system is a fixed point in Earth (e.g., WGS84) coordinates. This point is known to the TACSIT system using the interface by means beyond the scope of the interface.	
EARTH_REFERENCED	This value signifies that the origin for the coordinate system is well-defined with respect to the Earth by the coordinate system. E.g. centre of the Earth for Earth-Centred Earth-Fixed or the WGS84 spheroid for WGS84	
ENTITY_REFERENCED	The origin of the coordinate system is defined by the position of the relevant entity.	

6.1.5 DistanceUnit (Enumeration)

Enumeration of the known Distance Units: International system (m), nautical mile and mile.

Connections

Connector	Source	Target	Notes
Aggregation distanceUnit Source -> Destination	DistanceUnit	CoordinateUnits	The distance unit used by the group. SI is the default.

Attributes

Attribute	Notes	Default
SI	International System: meter.	
NauticalMile	Nautical Mile: "A nautical mile (symbol M, NM or nmi) is a unit of distance, set by international agreement as being exactly 1,852 meters (about 6,076 feet)." (https://en.wikipedia.org/wiki/Nautical_mile)	
Mile	Mile. The mile is an English unit of length equal to 1,760 yards and standardized as exactly 1,609.344 meters by international agreement in 1959. (https://en.wikipedia.org/wiki/Mile)	

6.1.6 Identity (Enumeration)

Identity according to STANAG 5516.

Attributes

Attribute	Notes	Default
PENDING	No identity assessment has yet been made	
UNKNOWN	The entity's identity is not yet determined	
ASSUMED_FRIEND	The entity is assumed to be friendly	
FRIEND	The entity is friendly	
NEUTRAL	The entity is neutral	
SUSPECT	The entity is considered suspicious	
HOSTILE	The entity is considered hostile	

6.1.7 SpeedUnit (Enumeration)

Enumeration of the known Speed Units: International system (m/s), nautical knot and mile per hour.

Connections

Connector	Source	Target	Notes
Aggregation speedUnit Source -> Destination	SpeedUnit	CoordinateUnits	The speed Unit used by the group. SI is the default.

Attributes

Attribute	Notes	Default
SI	International System: meter per second.	
Knot	The knot is a unit of speed equal to one nautical mile (1.852 km) per hour, approximately 1.151 mph. (https://en.wikipedia.org/wiki/Knot_%28unit%29)	
MilesPerHour	Miles per hour is an imperial and United States customary unit of speed expressing the number of statute miles covered in one hour. (https://en.wikipedia.org/wiki/Miles_per_hour) The English statute mile was established by a Weights and Measures Act of Parliament in 1593 during the reign of Queen Elizabeth I. (https://en.wikipedia.org/wiki/Mile#Statute_mile) 1 mph = 0.44704 m/s	

6.1.8 Environment (Enumeration)

The sensor tracking environment

Attributes

Attribute	Notes	Default
AIR	The entity is in the air environment	
LAND	The entity is on land	
SURFACE	The entity is on the surface of the sea	
SUBSURFACE	The entity is under the sea	
SPACE	The entity is in space	

6.1.9 TrackPhase (Enumeration)

The detection lifecycle phase of the track

Attributes

Attribute	Notes	Default
DEAD_RECKONED	Track provided based on extrapolated position (dead-reckoned)	
DELETED	Track has been deleted.	
LOST	Track has been lost	
TRACKED	Regular update of new and existing track	

6.1.10 Angle (DataType)

An angle is the measure of a rotation. The actual unit of this measure is specified elsewhere (see AngleUnit).

Implementation details are postponed to PSM-level choices.

6.1.11 DateTime (DataType)

DateTime is a specific time.

Implementation details (e.g. ISO 8601) are postponed to PSM-level choices.

6.1.12 Distance (DataType)

A distance is a measure of length. The actual unit of this measure is specified elsewhere (see DistanceUnit).

Implementation details are postponed to PSM-level choices.

6.1.13 Period (DataType)

A Period is the meantime between two times; it is specified as a couple of DateTime.

6.1.14 Projection (DataType)

A map projection is a systematic transformation of the latitudes and longitudes of locations on the surface of a sphere or an ellipsoid into locations on a plane. It is specified as a string naming the projection used, e.g. WGS84.

Connections

Connector	Source	Target	Notes
Aggregation projection Source -> Destination	Projection	CoordinateUnits	The Projection used by the group. WGS84 is the default.

6.1.15 Speed (DataType)

A speed is a measure of how fast something moves. The actual unit of this measure is specified elsewhere (see SpeedUnit).

Implementation details are postponed to PSM-level choices.

6.1.16 String (DataType)

A string is a sequence of characters.

Implementation details (such as UTF-8 encoding and so forth) are postponed to PSM-level choices.

6.1.17 URI (DataType)

An URI (Uniform Resource Identifier) is used to identify resources (entity, groups...). It is to be formatted as defined by RFC3986.

Implementation details (such as size limitation) are postponed to PSM-level choices.

6.1.18 URL (DataType)

An URL (Uniform Resource Locator) is used to reference resources. It is to be formatted as defined by RFC3986.

Implementation details (such as size limitation) are postponed to PSM-level choices.

6.2 GroupPayload

GroupPayload is the package of the classes needed to define the data exchanged with a TACSIT system for groups of entities.

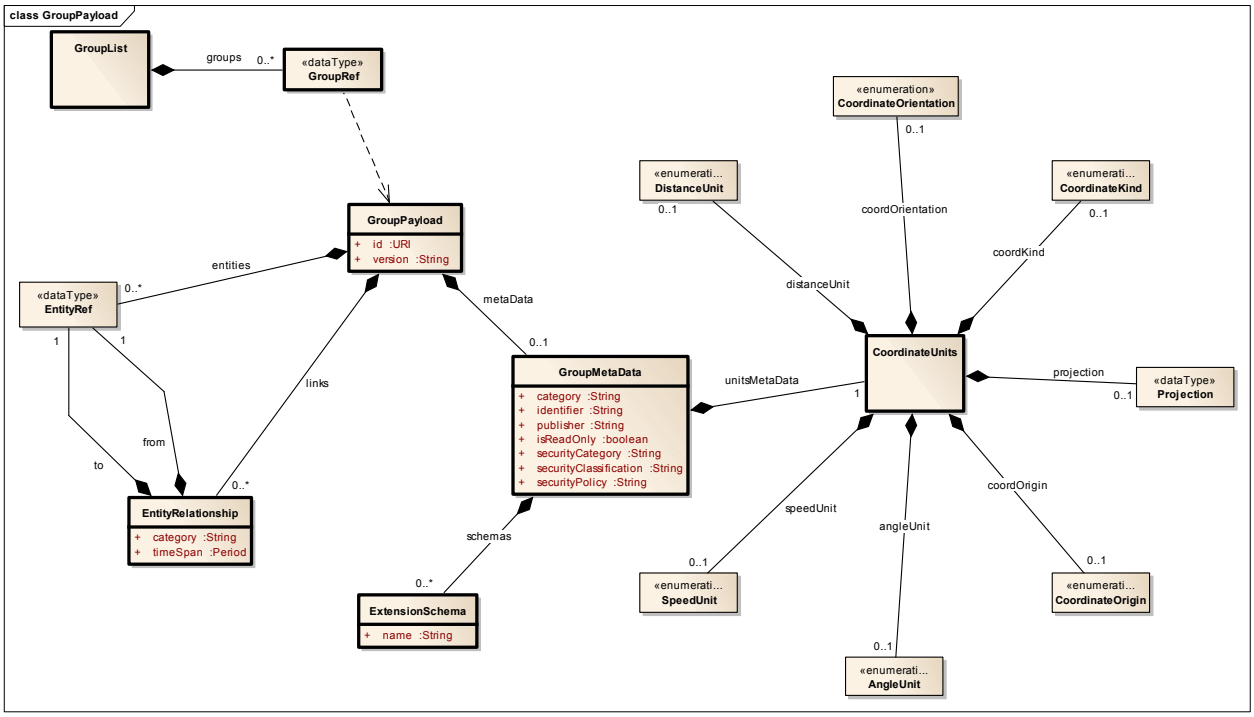


Figure 6.3 - GroupPayload (Class diagram)

6.2.1 CoordinateUnits (Class)

This class contains the metadata that are specific to units: unit of distance, unit of angle and so forth.

Connections

Connector	Source	Target	Notes
Aggregation angleUnit Source -> Destination	AngleUnit	CoordinateUnits	The angle unit used by the group. RAD is the default.
Aggregation coordKind Source -> Destination	CoordinateKind	CoordinateUnits	The coordinate kind unit used by the group. GEODETIC is the default.
Aggregation coordOrientation Source -> Destination	CoordinateOrientation	CoordinateUnits	The coordinate orientation used by the group. Default depend upon the coordinate kind.
Aggregation coordOrigin Source -> Destination	CoordinateOrigin	CoordinateUnits	The coordinate Origin used by the group. ABSOLUTE_REFERENCE_POINT is the default.
Aggregation distanceUnit	DistanceUnit	CoordinateUnits	The distance unit used by the

Source -> Destination			group. SI is the default.
Aggregation projection Source -> Destination	Projection	CoordinateUnits	The Projection used by the group. WGS84 is the default.
Aggregation speedUnit Source -> Destination	SpeedUnit	CoordinateUnits	The speed Unit used by the group. SI is the default.
Aggregation unitsMetaData Source -> Destination	CoordinateUnits	GroupMetaData	The metadata for the units: distance, angle and so forth.

6.2.2 EntityRelationship (Class)

An item of relationship from an entity to another one. This item is moreover specified by a category and lives within a time span.

Connections

Connector	Source	Target	Notes
Aggregation from Source -> Destination	EntityRef	EntityRelationship	The 'from' side of a relationship.
Aggregation links Source -> Destination	EntityRelationship	GroupPayload	List of relationships among Entities owned by the group.
Aggregation to Source -> Destination	EntityRef	EntityRelationship	The 'to' side of a relationship.

Attributes

Attribute	Notes	Default
category String	Category of the relationship. This is a String whose content is to be specified by the implementation.	
timeSpan Period	A show/hide period.	

6.2.3 ExtensionSchema (Class)

An extension schema is a namespace for the keys used in the extended data of an entity.

Annex A standardizes a first set of extension schemas.

Connections

Connector	Source	Target	Notes

Association schema Source -> Destination	ExtendedData	ExtensionSchema	Optional namespace of the key of an extended data.
Aggregation schemas Source -> Destination	ExtensionSchema	GroupMetaData	List of the extension schema that may be used to define the key of the extended data of the entities of the group.

Attributes

Attribute	Notes	Default
name String	Name of the schema.	

6.2.4 GroupList (Class)

A list of groups of entities. This is actually a list of references to groups.

Connections

Connector	Source	Target	Notes
Aggregation groups Source -> Destination	GroupRef	GroupList	List of grouped groups.

6.2.5 GroupMetaData (Class)

The Meta Data of a group: these metadata apply by default to all the entities of the group.

These data include the units which are used in all the attributes of all the entities of the group.

Connections

Connector	Source	Target	Notes
Aggregation metaData Source -> Destination	GroupMetaData	GroupPayload	the (optional) metadata of a group.
Aggregation schemas Source -> Destination	ExtensionSchema	GroupMetaData	List of the extension schema that may be used to define the key of the extended data of the entities of the group.
Aggregation unitsMetaData Source -> Destination	CoordinateUnits	GroupMetaData	The metadata for the units: distance, angle and so forth.

Attributes

Attribute	Notes	Default
category String	Category of the group of entities. This is a	

	String whose content is to be specified by the implementation.	
identifier String	An external identifier for the group. This identifier is not bound to be unique among groups.	
publisher String	The publisher of the group.	
isReadOnly boolean	State if the group is read only.	
securityCategory String	Description of the security classification (e.g. "Releasable for Internet transmission").	
securityClassification String	Security classification in the preceding policy (e.g. UNCLASSIFED, NATO COSMIC SECRET).	
securityPolicy String	Type of security Policy (e.g.: NATO).	

6.2.6 GroupPayload (Class)

The definition of a group of entities as it appears in the exchanges with TACSIT.

Note: the 'id' attribute is used by the TACSIT system to identify the GroupPayload when the 'identifier' attribute of the GroupMetatData is used by the system which requested the creation of this GroupPayload to identity internally it. 'identifier' is so a way to let this using system to find back its data when getting back the group from TACSIT.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	GroupRef	GroupPayload	
Aggregation entities Source -> Destination	EntityRef	GroupPayload	List of references to Entities owned by the group.
Aggregation links Source -> Destination	EntityRelationship	GroupPayload	List of relationships among Entities owned by the group.
Aggregation metaData Source -> Destination	GroupMetaData	GroupPayload	the (optional) metadata of a group.
Aggregation updatedGroup Source -> Destination	GroupPayload	GroupChangeSinkEvent	The created or modified group.
Aggregation	GroupPayload	GroupChangeEvent	The created or modified group.

updatedGroup Source -> Destination			

Attributes

Attribute	Notes	Default
id URI	A Uniform Resource Identifier (URI) that uniquely identifies the object.	
version String	Version of the group.	

6.2.7 GroupRef (DataType)

A GroupRef is a reference to a group. Its actual representation depends upon the PSM used.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	GroupRef	GroupPayload	
Aggregation deleted Source -> Destination	GroupRef	GroupChangeEvent	The deleted group.
Aggregation deleted Source -> Destination	GroupRef	GroupChangeSinkEvent	The deleted group.
Aggregation groups Source -> Destination	GroupRef	GroupList	List of grouped groups.

6.3 EntityPayload

EntityPayload is the package of the classes needed to define the data exchanged with a TACSIT system for entities.

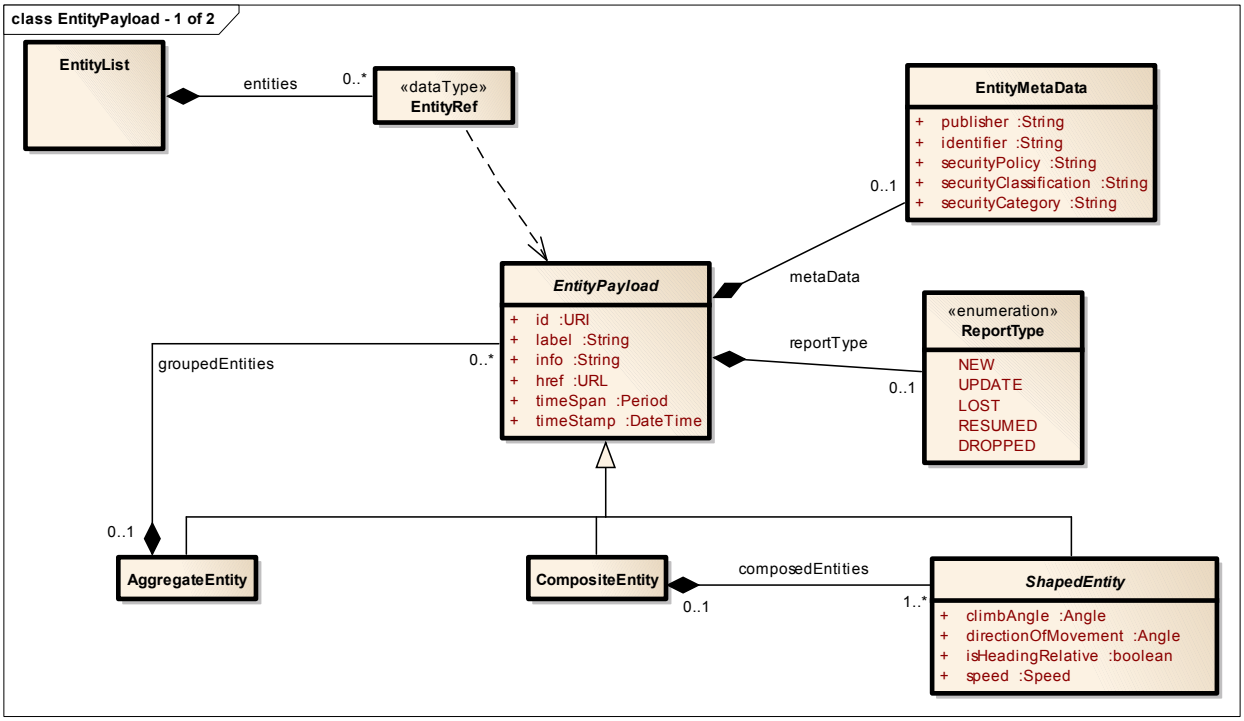


Figure 6.4 - EntityPayload - 1 of 2 (Class diagram)

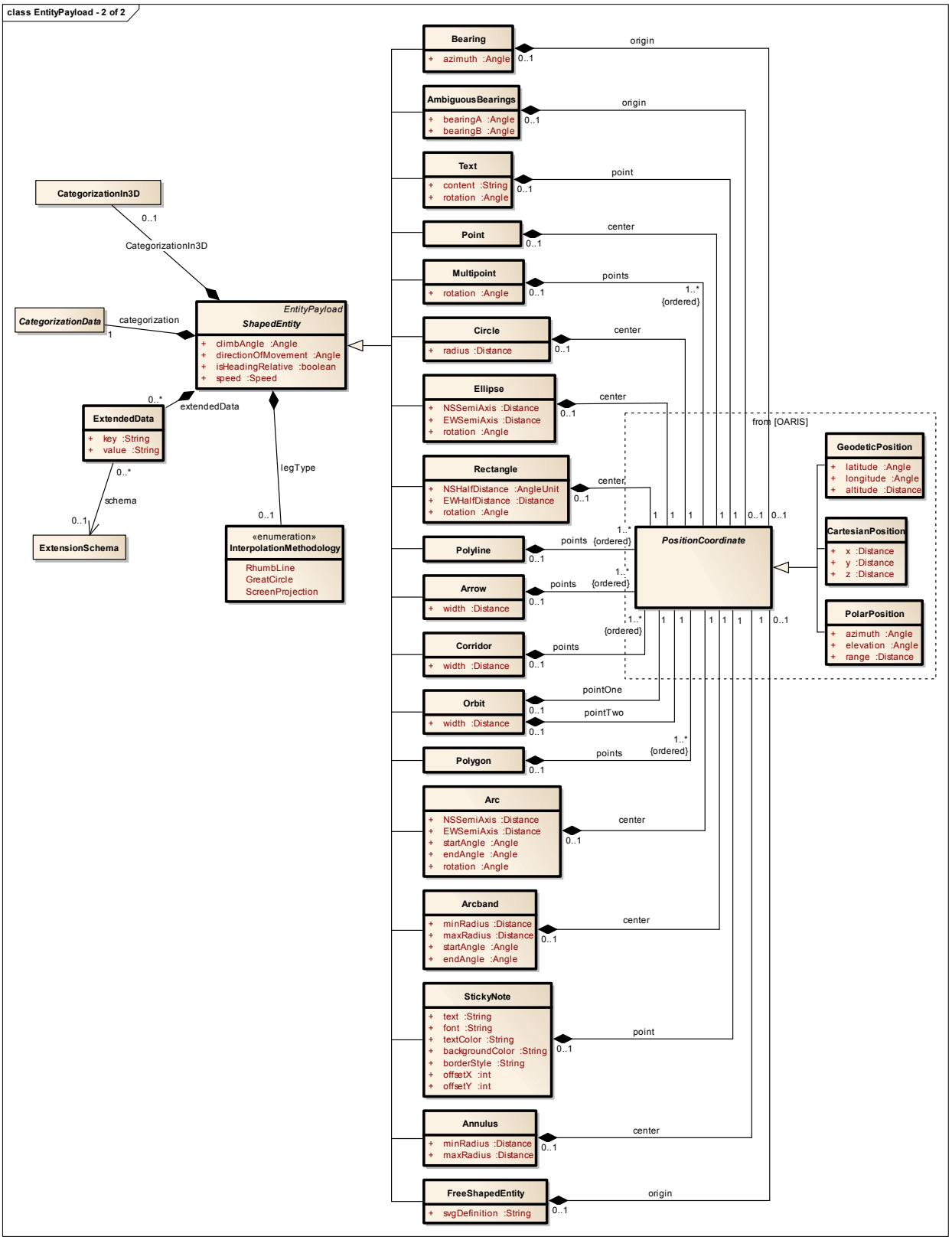


Figure 6.5 - EntityPayload - 2 of 2 (Class diagram)

6.3.1 AggregateEntity (Class)

Content Grouping.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	AggregateEntity	EntityPayload	
Aggregation groupedEntities Source -> Destination	EntityPayload	AggregateEntity	List of entities in the group.

6.3.2 AmbiguousBearings (Class)

An entity that is know to exist somewhere along a line of one of two angles of azimuth from an origin (typically the location of a passive towed array sonar), but there is no information regarding its range (distance from the origin). On a TACSIT each ambiguous bearing should be represented as a line from the origin to the edge of the display.

.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	AmbiguousBearings	ShapedEntity	
Aggregation origin Source -> Destination	PositionCoordinate	AmbiguousBearings	The origin of the ambiguous bearings.

Attributes

Attribute	Notes	Default
bearingA Angle	one of the ambiguous angles of azimuth	
bearingB Angle	the other ambiguous angle of azimuth	

6.3.3 Annulus (Class)

An entity represented as an area between two concentric circles and two radials of those circles. The annulus is defined by the two radii of the circles.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Annulus	ShapedEntity	
Aggregation center	PositionCoordinate	Annulus	The center of the annulus.

Source -> Destination			

Attributes

Attribute	Notes	Default
minRadius Distance	The radius of the smaller circle. This number should be positive The Distance unit is set by the Group.	
maxRadius Distance	The radius of the larger circle. This number should be positive The Distance unit is set by the Group.	

6.3.4 Arc (Class)

An entity represented as a segment of the outline of an ellipse. It is defined by the ellipse it is part of and the start and end angle of the arc on that ellipse. The arc is defined in a clockwise direction from the start angle to the end angle.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Arc	ShapedEntity	
Aggregation center Source -> Destination	PositionCoordinate	Arc	The center of the arc.

Attributes

Attribute	Notes	Default
NSSemiAxis Distance	The North-South (before rotation) semi-axis of the ellipse. This number should be positive. The Distance unit is set by the Group.	
EWSemiAxis Distance	The East-West (before rotation) semi-axis of the ellipse. This number should be positive. The Distance unit is set by the Group.	
startAngle Angle	The start angle of the arc along an ellipse prior to rotation. The unit is set by the group.	
endAngle Angle	The end angle of the arc along an ellipse prior to rotation. The unit is set by the group.	
rotation Angle	Rotation in the counter-clockwise direction. The unit is set by the group.	

6.3.5 Arcband (Class)

An entity represented as an area between two concentric circles and two radials of those circles. The arcband is defined by the two radii of the circles and the two angles of the radials moving from startangle to endangle in a clockwise direction.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Arcband	ShapedEntity	
Aggregation center Source -> Destination	PositionCoordinate	Arcband	The center of the arcband.

Attributes

Attribute	Notes	Default
minRadius Distance	The radius of the smaller circle. This number should be positive The Distance unit is set by the Group.	
maxRadius Distance	The radius of the smaller circle. This number should be positive The Distance unit is set by the Group.	
startAngle Angle	The start angle of the arc along an ellipse prior to rotation. The unit is set by the group.	
endAngle Angle	The end angle of the arc along an ellipse prior to rotation. The unit is set by the group.	

6.3.6 Arrow (Class)

An entity represented as a polyline.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Arrow	ShapedEntity	
Aggregation points Source -> Destination	PositionCoordinate ordered	Arrow	The points that make up the arrow. The order of points defines the direction proceeding from tail to point.

Attributes

Attribute	Notes	Default
width Distance	Width of the arrow body. Width must be greater than zero. The Distance unit is set by the Group.	

6.3.7 Bearing (Class)

An entity that is know to exist somewhere along a line of azimuth from an origin (typically the location of a passive sensor), but there is no information regarding its range (distance from the origin). On a TACSIT a bearing should be represented as a line from the origin to the edge of the display.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Bearing	ShapedEntity	
Aggregation origin Source -> Destination	PositionCoordinate	Bearing	The origin of the bearing.

Attributes

Attribute	Notes	Default
azimuth Angle	The angle of azimuth for the bearing	

6.3.8 CartesianPosition (Class)

Coordinates in a Cartesian reference frame as described by a coordinate specification object (from [OARIS]).

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	CartesianPosition	PositionCoordinate	

Attributes

Attribute	Notes	Default
x Distance	X position of the point. The Distance unit is set by the Group.	
y Distance	Y position of the point. The Distance unit is set by the Group.	
z Distance	Altitude of the point. The Distance unit is set by the Group.	

6.3.9 Circle (Class)

An entity represented as a circle.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Circle	ShapedEntity	
Aggregation center Source -> Destination	PositionCoordinate	Circle	The coordinates of the center of the circle.

Attributes

Attribute	Notes	Default
radius Distance	Radius of the center of the circle. The Distance unit is set by the Group.	

6.3.10 CompositeEntity (Class)

Content unbreakable composition made of basic shapes (no recursion).

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	CompositeEntity	EntityPayload	
Aggregation composedEntities Source -> Destination	ShapedEntity	CompositeEntity	List of entities in the composition.

6.3.11 Corridor (Class)

An entity represented as a corridor.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Corridor	ShapedEntity	
Aggregation points Source -> Destination	PositionCoordinate ordered	Corridor	The waypoints of the corridor.

Attributes

Attribute	Notes	Default
width Distance	Width of the Corridor. Width must be greater than zero. The Distance unit is set by the Group.	

6.3.12 Ellipse (Class)

An entity represented as an ellipse.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Ellipse	ShapedEntity	
Aggregation center Source -> Destination	PositionCoordinate	Ellipse	The coordinates of the center of the ellipse.

Attributes

Attribute	Notes	Default
NSSemiAxis Distance	The North-South (before rotation) semi-axis of the ellipse. This number should be positive. The Distance unit is set by the Group.	
EWSemiAxis Distance	The East-West (before rotation) semi-axis of the ellipse. This number should be positive. The Distance unit is set by the Group.	
rotation Angle	Rotation in the counter-clockwise direction. The unit is set by the group.	

6.3.13 EntityList (Class)

A list of entities. This is actually a list of references to entities.

Connections

Connector	Source	Target	Notes
Aggregation entities Source -> Destination	EntityRef	EntityList	The grouped entities.

6.3.14 EntityMetaData (Class)

The Meta Data of an Entity.

Connections

Connector	Source	Target	Notes
Aggregation metaData	EntityMetaData	EntityPayload	Metadata for the entity. These metadata, if present, replace the

Source -> Destination			ones of the group.

Attributes

Attribute	Notes	Default
publisher String	The publisher of the entity.	
identifier String	An external identifier for the entity. This identifier is not bound to be unique among entities.	
securityPolicy String	Type of security Policy (e.g.: NATO).	
securityClassification String	Security classification in the preceding policy (e.g. UNCLASSIFED, NATO COSMIC SECRET).	
securityCategory String	Description of the security classification (e.g. "Releasable for Internet transmission").	

6.3.15 EntityPayload (Class)

The definition of an entity as it appears in the exchanges with TACSIT.

The 'reportType' attributes gives the lifecycle status of this payload showing that it is a creation, a modification and so forth.

Note: the 'id' attribute is used by the TACSIT system to identify the EntityPayload while the 'identifier' attribute of the EntityMetatData is used by the system which requested the creation of this EntityPayload to identify it internally. 'identifier' is so a way to let this using system to find back its data when getting back an entity from TACSIT.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	AggregateEntity	EntityPayload	
Generalization Source -> Destination	CompositeEntity	EntityPayload	
Dependency Source -> Destination	EntityRef	EntityPayload	
Generalization Source -> Destination	ShapedEntity	EntityPayload	
Aggregation groupedEntities	EntityPayload	AggregateEntity	List of entities in the group.

Source -> Destination			
Aggregation metaData Source -> Destination	EntityMetaData	EntityPayload	Metadata for the entity. These metadata, if present, replace the ones of the group.
Aggregation reportType Source -> Destination	ReportType	EntityPayload	Lifecycle status of the EntityPayload. Default is UPDATE.
Aggregation states Source -> Destination	EntityPayload	EntityHistoryPayload	The list of time-stamped states of the history.
Aggregation updatedEntity Source -> Destination	EntityPayload	EntityChangeEvent	The created or modified entity.

Attributes

Attribute	Notes	Default
id URI	A Uniform Resource Identifier (URI) that uniquely identifies the object.	
label String	Short description.	
info String	Additional human-readable text.	
href URL	A URL to human readable content providing more information about the object.	
timeSpan Period	A show/hide period.	
timeStamp DateTime	The date of validity of the information held by the entity.	

6.3.16 ExtendedData (Class)

This class allows for extended data for an entity. An extended data is defined by a schema/key/value triple, the schema being a group of keys (sort of namespace).

Extended Data are typically used to holds business-specific data that may be drawn/written around a symbol.

Annex A standardizes a first set of extended data keys.

Connections

Connector	Source	Target	Notes
Aggregation extendedData Source -> Destination	ExtendedData	ShapedEntity	Extensible data.
Association schema	ExtendedData	ExtensionSchema	Optional namespace of the key of an extended data.

Source -> Destination			

Attributes

Attribute	Notes	Default
key String	Key of the extended data to be taken in the schema.	
value String	Value matching the key.	

6.3.17 FreeShapedEntity (Class)

An entity represented in a client specific format through the SVG (Scalable Vector Graphics) standard.

It has an optional origin. If it contains an origin then the SVG coordinates are interpreted as Cartesian relative to that origin. If there is no origin then the coordinates are interpreted with respect to a related entity (from an Entity Relation) as Cartesian. If there is no related entity then the coordinates are interpreted as absolute latitudes (y) and longitudes (x).

This class supports the display of geometry for entities for which the TACSIT has no pre-defined symbology standard.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	FreeShapedEntity	ShapedEntity	
Aggregation origin Source -> Destination	PositionCoordinate	FreeShapedEntity	The origin by which to interpret the coordinates within the SVG free shape definition.

Attributes

Attribute	Notes	Default
svgDefinition String	An SVG xml file defining the appearance of the free shaped entity.	

6.3.18 GeodeticPosition (Class)

Coordinates in a Geodetic reference frame as described by a coordinate specification object (from [OARIS]).

The datum (WGS84...) is set by the group.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	GeodeticPosition	PositionCoordinate	

Attributes

Attribute	Notes	Default
latitude Angle	Latitude of the point.	

	The Angle unit as well as the projection are set by the Group.	
longitude Angle	Longitude of the point. The Angle unit as well as the projection are set by the Group.	
altitude Distance	Altitude of the point. The Distance unit as well as the projection are set by the Group.	

6.3.19 InterpolationMethodology (Enumeration)

Interpolation of the line positions between two points is performed using one of the following methodologies:

- Rhumb Line – constant heading;

- Great Circle – shortest path;

- ScreenProjection - Screen.

Connections

Connector	Source	Target	Notes
Aggregation legType Source -> Destination	InterpolationMethodology	ShapedEntity	Interpolation method for the shape.

Attributes

Attribute	Notes	Default
RhumbLine	Interpolation along a straight line on a Mercator projection chart.	
GreatCircle	Interpolation along the surface of the sphere.	
ScreenProjection	Interpolation along a straight line on the screen.	

6.3.20 Multipoint (Class)

A multipoint is a list of points without the semantics of a polyline (each point must be linked with a line) or a polygon (each point must be linked with a line and the figure is closed). The semantics is given by the associated CategorizationData. As examples, 2525 and APP-6 series make a clear difference between multipoint and line.

The rotation says that once the figure is drawn according to the CategorizationData, it must be rotated.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Multipoint	ShapedEntity	
Aggregation points Source -> Destination	PositionCoordinate ordered	Multipoint	The coordinates of the mulitpoints

Attributes

Attribute	Notes	Default
rotation Angle	Rotation in the counter-clockwise direction. The unit is set by the group.	

6.3.21 Orbit (Class)

An entity represented as an orbit path.

The intended result the shape formed by linking together two half circles with two lines (a.k.a. an athletics track by the sportsmen among us). The first circle is centered on point one and the second circle is centered on point two. Both of them has the same radius given by the width attribute.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Orbit	ShapedEntity	
Aggregation pointOne Source -> Destination	PositionCoordinate	Orbit	First of the 2 points of the supporting corridor defining the sides of the orbit. The orbit is centered on the line across these 2 points.
Aggregation pointTwo Source -> Destination	PositionCoordinate	Orbit	Second of the 2 points of the supporting corridor defining the sides of the orbit. The orbit is centered on the line across these 2 points.

Attributes

Attribute	Notes	Default
width Distance	Width of the orbit. Width must be greater than zero. The Distance unit is set by the Group.	

6.3.22 Point (Class)

An entity represented as a point.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Point	ShapedEntity	

Aggregation center Source -> Destination	PositionCoordinate	Point	The coordinates of the point.

6.3.23 PolarPosition (Class)

Coordinates in a polar reference frame as a described by a coordinate specification object (from [OARIS]).

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	PolarPosition	PositionCoordinate	

Attributes

Attribute	Notes	Default
azimuth Angle	Azimuth of the point. The Angle unit is set by the Group.	
elevation Angle	Elevation of the point. The Angle unit is set by the Group.	
range Distance	Distance of the point from the center. The Distance unit as well as the projection are set by the Group.	

6.3.24 Polygon (Class)

An entity represented as a polygon.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Polygon	ShapedEntity	
Aggregation points Source -> Destination	PositionCoordinate ordered	Polygon	The points of the polygon

6.3.25 Polyline (Class)

An entity represented as a polyline.

Connections

Connector	Source	Target	Notes
Generalization	Polyline	ShapedEntity	

Source -> Destination			
Aggregation points Source -> Destination	PositionCoordinate ordered	Polyline	The coordinates of the points of the polyline

6.3.26 PositionCoordinate (Class)

A georeferenced point.

The type of coordinate (WGS84/Cartesian/Polar), the type of orientation as well as the type of origin are set by the group.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	PolarPosition	PositionCoordinate	
Generalization Source -> Destination	GeodeticPosition	PositionCoordinate	
Generalization Source -> Destination	CartesianPosition	PositionCoordinate	
Aggregation center Source -> Destination	PositionCoordinate	Annulus	The center of the annulus.
Aggregation center Source -> Destination	PositionCoordinate	Point	The coordinates of the point.
Aggregation center Source -> Destination	PositionCoordinate	Circle	The coordinates of the center of the circle.
Aggregation center Source -> Destination	PositionCoordinate	Arc	The center of the arc.
Aggregation center Source -> Destination	PositionCoordinate	Rectangle	The coordinates of the center of the rectangle.
Aggregation center	PositionCoordinate	Arcband	The center of the arcband.

Source -> Destination			
Aggregation center Source -> Destination	PositionCoordinate	Ellipse	The coordinates of the center of the ellipse.
Aggregation origin Source -> Destination	PositionCoordinate	AmbiguousBearings	The origin of the ambiguous bearings.
Aggregation origin Source -> Destination	PositionCoordinate	FreeShapedEntity	The origin by which to interpret the coordinates within the SVG free shape definition.
Aggregation origin Source -> Destination	PositionCoordinate	Bearing	The origin of the bearing.
Aggregation point Source -> Destination	PositionCoordinate	StickyNote	Location of the end of the handle.
Aggregation point Source -> Destination	PositionCoordinate	Text	Position of the text (see description).
Aggregation pointOne Source -> Destination	PositionCoordinate	Orbit	First of the 2 points of the supporting corridor defining the sides of the orbit. The orbit is centered on the line across these 2 points.
Aggregation points Source -> Destination	PositionCoordinate ordered	Polygon	The points of the polygon
Aggregation points Source -> Destination	PositionCoordinate ordered	Arrow	The points that make up the arrow. The order of points defines the direction proceeding from tail to point.
Aggregation points Source -> Destination	PositionCoordinate ordered	Corridor	The waypoints of the corridor.
Aggregation points Source -> Destination	PositionCoordinate ordered	Polyline	The coordinates of the points of the polyline

Aggregation points Source -> Destination	PositionCoordinate ordered	Multipoint	The coordinates of the mulitpoints
Aggregation pointTwo Source -> Destination	PositionCoordinate	Orbit	Second of the 2 points of the supporting corridor defining the sides of the orbit. The orbit is centered on the line across these 2 points.

6.3.27 Rectangle (Class)

An entity represented as a rectangle.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Rectangle	ShapedEntity	
Aggregation center Source -> Destination	PositionCoordinate	Rectangle	The coordinates of the center of the rectangle.

Attributes

Attribute	Notes	Default
NSHalfDistance AngleUnit	The dimension from center point to edge (half distance) of the rectangle along its North-South length. This number should be positive. The Distance unit is set by the Group.	
EWHalfDistance Distance	The dimension from center point to edge (half distance) of the rectangle along its East-West length. This number should be positive. The Distance unit is set by the Group.	
rotation Angle	Rotation in the counter-clockwise direction. The unit is set by the group.	

6.3.28 ReportType (Enumeration)

Enumeration of the lifecycle status for an EntityPayload.

Connections

Connector	Source	Target	Notes
Aggregation reportType Source -> Destination	ReportType	EntityPayload	Lifecycle status of the EntityPayload.

			Default is UPDATE.

Attributes

Attribute	Notes	Default
NEW	State that the EntityPayload has been created.	
UPDATE	State that the EntityPayload has been modified.	
LOST	State that the EntityPayload has been lost.	
RESUMED	State that the EntityPayload has been resumed (after lost).	
DROPPED	State that an EntityPayload has been dropped from the system.	

6.3.29 ShapedEntity (Class)

The base for most items.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	FreeShapedEntity	ShapedEntity	
Generalization Source -> Destination	Rectangle	ShapedEntity	
Generalization Source -> Destination	Text	ShapedEntity	
Generalization Source -> Destination	Bearing	ShapedEntity	
Generalization Source -> Destination	Circle	ShapedEntity	
Generalization Source -> Destination	Point	ShapedEntity	
Generalization Source -> Destination	Arc	ShapedEntity	

Generalization Source -> Destination	Arrow	ShapedEntity	
Generalization Source -> Destination	Polygon	ShapedEntity	
Generalization Source -> Destination	Polyline	ShapedEntity	
Generalization Source -> Destination	ShapedEntity	EntityPayload	
Generalization Source -> Destination	StickyNote	ShapedEntity	
Generalization Source -> Destination	AmbiguousBearings	ShapedEntity	
Generalization Source -> Destination	Orbit	ShapedEntity	
Generalization Source -> Destination	Annulus	ShapedEntity	
Generalization Source -> Destination	Corridor	ShapedEntity	
Generalization Source -> Destination	Arcband	ShapedEntity	
Generalization Source -> Destination	Ellipse	ShapedEntity	
Generalization Source -> Destination	Multipoint	ShapedEntity	
Aggregation	CategorizationData	ShapedEntity	Data needed to draw the symbol of

categorization Source -> Destination			the entity.
Aggregation CategorizationIn3D Source -> Destination	CategorizationIn3D	ShapedEntity	Data needed to draw the symbol of the entity in 3D.
Aggregation composedEntities Source -> Destination	ShapedEntity	CompositeEntity	List of entities in the composition.
Aggregation extendedData Source -> Destination	ExtendedData	ShapedEntity	Extensible data.
Aggregation legType Source -> Destination	InterpolationMethodology	ShapedEntity	Interpolation method for the shape.

Attributes

Attribute	Notes	Default
climbAngle Angle	Direction of the movement wrt altitude, if any. The unit is set by the group.	
directionOfMovement Angle	Direction of the movement, if any. The unit is set by the group.	
isHeadingRelative boolean	Whether the shape is defined relative to the heading (direction of movement) of the entity	
speed Speed	Speed of movement. The unit is set by the group.	

6.3.30 StickyNote (Class)

A note with a handle.

Also know as a Post-It note ("Post-It" is a registered 3M trademark).

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	StickyNote	ShapedEntity	
Aggregation point Source -> Destination	PositionCoordinate	StickyNote	Location of the end of the handle.

Attributes

Attribute	Notes	Default
text String	Text of the note	
font String	Font family (aka typeface) used to write the text within the note.	
textColor String	Color of the text in the note. The way this color is mapped to RGB or other values is implementation dependent.	
backgroundColor String	Color of the note itself. The way this color is mapped to RGB or other values is implementation dependent.	
borderStyle String	Style of the border of the note. The way this style is mapped is implementation dependent.	
offsetX int	Horizontal gap between the center of the text block and the end of the handle. In pixels.	
offsetY int	Vertical gap between the center of the text block and the end of the handle. In pixels.	

6.3.31 Text (Class)

An entity represented as a text.

The 'point' attribute gives the left-lower starting position for left-to-right top-to-bottom languages, the right-lower starting position for right-to-left top-to-bottom languages, the left-upper starting position for top-to-bottom left-to-right languages and the right-upper starting position for top-to-bottom right-to-left languages.

This attribute is implementation dependent for the other languages.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Text	ShapedEntity	
Aggregation point Source -> Destination	PositionCoordinate	Text	Position of the text (see description).

Attributes

Attribute	Notes	Default
content String	The text to be displayed.	
rotation Angle	Rotation in the counter-clockwise direction. The unit is set by the group.	

6.3.32 EntityRef (DataType)

An EntityRef is a reference to an EntityPayload. Its actual representation depends upon the PSM used.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	EntityRef	EntityPayload	
Aggregation deleted Source -> Destination	EntityRef	EntityChangeEvent	The deleted entity.
Aggregation deleted Source -> Destination	EntityRef	EntityChangeSinkEvent	The deleted entity.
Aggregation entities Source -> Destination	EntityRef	EntityList	The grouped entities.
Aggregation entities Source -> Destination	EntityRef	GroupPayload	List of references to Entities owned by the group.
Aggregation from Source -> Destination	EntityRef	EntityRelationship	The 'from' side of a relationship.
Aggregation reference Source -> Destination	EntityRef	EntityHistoryPayload	The entity concerned by the history.
Aggregation to Source -> Destination	EntityRef	EntityRelationship	The 'to' side of a relationship.

6.4 CategorizationData

CategorizationData are data needed to draw an Entity depending on the chosen symbology: 2525, APP-6...

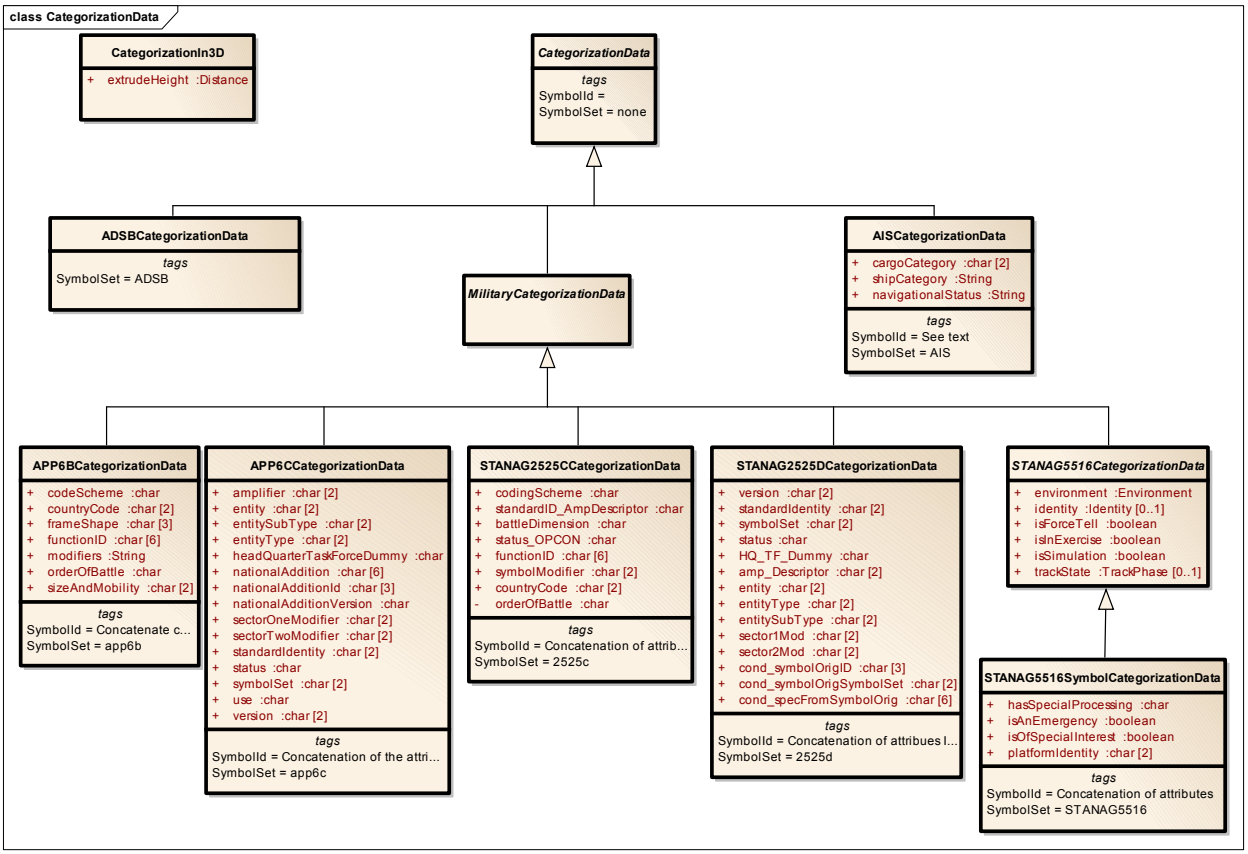


Figure 6.6 - CategorizationData (Class diagram)

6.4.1 ADSBCategorizationData (Class)

This class encapsulates data specifically received from ADS-B broadcast.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	ADSBCategorizationData	CategorizationData	

6.4.2 AISCategorizationData (Class)

The data needed for displaying the AIS symbol set.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	AISCategorizationData	CategorizationData	

Attributes

Attribute	Notes	Default
cargoCategory char	The IMO categorization of the cargo carried as identified on AIS	
shipCategory String	The type of ship.	
navigationalStatus String	Description of current navigational status and readiness	

6.4.3 APP6BCategorizationData (Class)

The data needed for displaying APP-6B symbol set (see [APP-6B]).

The method to work out the symbol id is to concatenate the attributes codeScheme, frameShape, functionID, sizeAndMobility, countryCode, orderOfBattle.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	APP6BCategorizationData	MilitaryCategorizationData	

Attributes

Attribute	Notes	Default
codeScheme char	Code scheme, position 1. This position indicates the overall symbology set to which a symbol belongs.	
countryCode char	Country code, positions 13 and 14. These positions identify the country with which a symbol is associated. Country code identifiers are listed in Federal Information Processing Standard Pub 10 series (See [FIPS 10-4]).	
frameShape char	Affiliation, battle dimension, and status, 2, 3, and 4. These positions determine the frame shape of a symbol and indicate its actual or planned location.	
functionID char	Function ID, positions 5 through 10. These positions identify a symbol's function, with each position providing increasing levels of detail and specialization.	
modifiers String	The semicolon-separated list of Symbol Modifier Fields as defined is [APP6B]. A Symbol Modifier Field is defined as a key-value pair delimited with the ':' character.	
orderOfBattle char	Order of battle, position 15. This position provides additional information about the role	

	of a symbol in the battlespace. For example, a bomber that has nuclear weapons on board may be strategic force-related, or a tactical graphic may also perform the role of a control point.	
sizeAndMobility char	Size/mobility indicator code, positions 11 and 12. These positions identify the size and mobility of a symbol.	

6.4.4 APP6CCategorizationData (Class)

The data needed for displaying APP-6C symbol set as described in [APP-6C].

The method to work out the symbol id is to concatenate the attributes.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	APP6CCategorizationData	MilitaryCategorizationData	

Attributes

Attribute	Notes	Default
amplifier char	The Unit Echelon/Equipment Mobility/Naval Towed Array Amplifier is comprised of two digits.	
entity char		
entitySubType char		
entityType char		
headQuarterTaskForceDummy char	The Headquarters/Task Force/Dummy is comprised of one digit.	
nationalAddition char	Specified by national or geopolitical symbol set. This is to accommodate national modifications/ additions that are not included in APP-6C.	
nationalAdditionId char	National or geopolitical identifier. This is to accommodate national modifications/ additions that are not included in APP-6C.	
nationalAdditionVersion char	National or geopolitical symbol set version. This is to accommodate national modifications/ additions that are not included in APP-6C.	
sectorOneModifier char		
sectorTwoModifier char		

standardIdentity char	Standard identity is comprised of two digits. The first digit represents the context of the symbol and the second digit reflects the standard identities.	
status char	The status is comprised of one digit.	
symbolSet char	The symbol set is comprised of two digits.	
use char	This one digit field indicates whether the symbol identification code uses the third ten digit set.	
version char	This field identifies a version change for the symbol identification code which occurs when there is a change in an established icon, modifier, or drawing rule for a control measure symbol. Subsequent changes will create further version changes for the symbol identification code. The basis for all symbol versions in all sets is APP-6(C).	

6.4.5 CategorizationData (Class)

The data of an entity used to choose the symbol used to draw the given entity.

Each leaf subclass must define a SymbolSet (as a String) and a SymbolId (as a method to work out a symbol id from the attributes) that may be used in the PSMs.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	ADSBCategorizationData	CategorizationData	
Generalization Source -> Destination	MilitaryCategorization Data	CategorizationData	
Generalization Source -> Destination	AISCategorizationData	CategorizationData	
Aggregation categorization Source -> Destination	CategorizationData	ShapedEntity	Data needed to draw the symbol of the entity.

6.4.6 CategorizationIn3D (Class)

The data of an entity used for the 3D rendering of this entity.

Connections

Connector	Source	Target	Notes
Aggregation CategorizationIn3D Source -> Destination	CategorizationIn3D	ShapedEntity	Data needed to draw the symbol of the entity in 3D.

Attributes

Attribute	Notes	Default
extrudeHeight Distance	Line and point symbols can be extruded above the terrain for visual emphasis, forming what appear to be walls on the terrain surface. This attribute gives the height of the lower border of this "wall".	

6.4.7 MilitaryCategorizationData (Class)

The military categorization data : APP-6, 2525...

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	APP6BCategorizationData	MilitaryCategorizationData	
Generalization Source -> Destination	MilitaryCategorizationData	CategorizationData	
Generalization Source -> Destination	STANAG2525CCategorizationData	MilitaryCategorizationData	
Generalization Source -> Destination	STANAG5516CategorizationData	MilitaryCategorizationData	
Generalization Source -> Destination	APP6CCategorizationData	MilitaryCategorizationData	
Generalization Source -> Destination	STANAG2525DCategorizationData	MilitaryCategorizationData	

6.4.8 STANAG2525CCategorizationData (Class)

The data needed for displaying the 2525C symbol set.

For a full definition of the specifications of using MILSTD-2525 Symbology refer to the published version from the US-DOD.

Excerpt from Appendix A of the 2525C Symbology Standard:

A.5.1 Technical specifications. Composition, construction, display, and transmission of tactical symbols are explained in the detailed requirements section of the standard.

A.5.2 Symbol identification coding scheme. A SIDC is a 15-character alphanumeric identifier that provides the information necessary to display or transmit a tactical symbol between MIL-STD-2525 compliant systems.

Apendix A of 2525C Specification outlines the order of concatenation of attribute data. Copied here for convenience

Attribute - # of Chars

Coding Scheme - 1 (S for Warfighting)

Standard ID - 1

Battle Dimension - 1

Status/OPCON -1

Function ID - 6

Symbol Modifier - 2

Country Code - 2

Order of Battle - 1

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	STANAG2525CCategorizationData	MilitaryCategorizationData	

Attributes

Attribute	Notes	Default
codingScheme char	Version: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
standardID_AmpDescripto	Standard Identity: See detailed specification of	

r char	this field in the Specification (MIL-STD-2525D Appendix A.5).	
battleDimension char	Symbol Set: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
status_OPCON char	Status: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
functionID char	HQ/Task Force/Dummy: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
symbolModifier char	Amplifier/Descriptor: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
countryCode char	Entity: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
orderOfBattle char		

6.4.9 STANAG2525DCategorizationData (Class)

The data needed for displaying the 2525D symbol set.

For a full definition of the specifications of using MILSTD-2525 Symbology refer to the latest published version from the US-DOD.

Excerpt from Appendix A of the 2525D Symbology Standard:

A.5.1 Symbol identification codes. A symbol identification code is a numeric code that uniquely identifies the elements needed to build a MIL–STD–2525D compliant symbol. The numeric codes provide the same type of descriptions used in message formats but further focus

the data to a specific domain for ease in creating the symbols with less band width.

A.5.2 Elements of the symbol identification codes. The symbol identification code is composed of eleven elements of information which are presented in two sets of ten digits. An additional set of ten digits composed of three elements must be used when a symbology

originator version extension flag is used. This extension is conditional.

Apendix A of 2525D Specification outlines the order of concatenation of attribute data. Copied here for convenience

Attribute - # of Chars

Set A:

Version - 2

Standard ID - 2

Symbol Set - 2

Status -1

HQ Task Force Dummy - 1

Amp / Descr - 2

Set B:

Entity - 2

Entity Type - 2

Entity Subtype - 2

Sector 1 Mod - 2

Sector 2 Mod - 2

Conditional Set C

Symbology Orig ID - 3

Symbology Orig Symbol Set - 1

Specified by Symbol Originator - 2,2,2

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	STANAG2525DCategorizationData	MilitaryCategorizationData	

Attributes

Attribute	Notes	Default
version char	Version: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
standardIdentity char	Standard Identity: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
symbolSet char	Symbol Set: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
status char	Status: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
HQ_TF_Dummy char	HQ/Task Force/Dummy: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
amp_Descriptor char	Amplifier/Descriptor: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
entity char	Entity: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
entityType char	Entity Type: See detailed specification of this field in the Specification (MIL-STD-2525D	

	Appendix A.5).	
entitySubType char	Entity Subtype: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
sector1Mod char	Sector 1 modifier: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
sector2Mod char	Sector 2 modifier: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
cond_symbolOrigID char	Conditional Set C Attribute - Symbology Originator ID: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
cond_symbolOrigSymbolSet char	Conditional Set C Attribute - Symbology Originator Symbol Set: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	
cond_specFromSymbolOrig char	Conditional Set C Attribute - Specified From Symbology Originator: See detailed specification of this field in the Specification (MIL-STD-2525D Appendix A.5).	

6.4.10 STANAG5516CategorizationData (Class)

This is a base class to support the display of entities based on the STANAG 5516 data model. This class contains attributes that relate to display filtering for any entities that have no symbol

Note that this STANAG does not define symbology. The purpose of this class is to support custom symbology based on 5516 data model.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	STANAG5516SymbolCategorizationData	STANAG5516CategorizationData	
Generalization Source -> Destination	STANAG5516CategorizationData	MilitaryCategorizationData	

Attributes

Attribute	Notes	Default
environment Environment	The environment that the Entity is in or to which it applies	
identity Identity	The standard identity of the (track-like) entity	

isForceTell boolean	The ForceTell indicator is set	
isInExercise boolean	The Entity has an Exercise indicator and so is part of an exercise.	
isSimulation boolean	The Simulation indicator is set - e.g. the entity is for operator training purposes.	
trackState TrackPhase	If a track-like entity the state or phase of its lifecycle. Indicates whether tracking is active	

6.4.11 STANAG5516SymbolCategorizationData (Class)

The data needed to display symbol sets aimed at data models derived from STANAG 5516. This STANAG does not define symbology. The purpose of this class is to support custom symbology based on 5516 data model.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	STANAG5516SymbolCategorizationData	STANAG5516CategorizationData	

Attributes

Attribute	Notes	Default
hasSpecialProcessing char	The Special Processing indicator is set	
isAnEmergency boolean	The Emergency indicator is set	
isOfSpecialInterest boolean	The Special Interest indicator is set	
platformIdentity char	The code for the per environment defined platform identify for the Entity according to STANAG 5516. If no identity is defined then the code refers to a value in the reference point tables within STANAG 5516.	

6.5 EntityHistory

EntityHistory is the package of the classes needed to define the data exchanged with a TACSIT system for histories of entities.

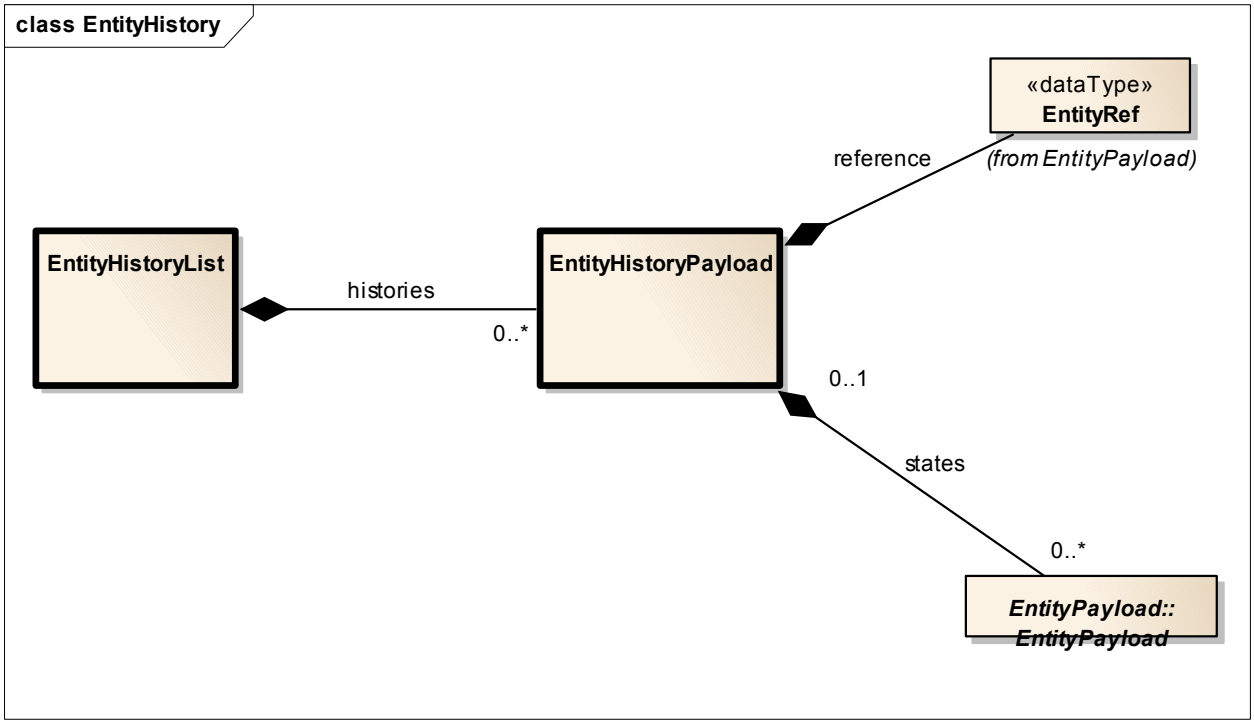


Figure 6.7 - EntityHistory (Class diagram)

6.5.1 EntityHistoryList (Class)

List of entity histories.

Connections

Connector	Source	Target	Notes
Aggregation histories Source -> Destination	EntityHistoryPayload	EntityHistoryList	The grouped histories

6.5.2 EntityHistoryPayload (Class)

The definition of the history of an entity as it appears in the exchanges with TACSIT.

Such a history is defined:

- For one entity;

- By a collection of Entity Payloads which are the different states in the history (as a reminder, an EntityPayload holds a TimeStamp attribute giving the date of validity of the data).

Connections

Connector	Source	Target	Notes
Aggregation histories Source -> Destination	EntityHistoryPayload	EntityHistoryList	The grouped histories

Aggregation reference Source -> Destination	EntityRef	EntityHistoryPayload	The entity concerned by the history.
Aggregation states Source -> Destination	EntityPayload	EntityHistoryPayload	The list of time-stamped states of the history.
Aggregation updatedHistory Source -> Destination	EntityHistoryPayload	HistoryChangeEvent	The created or modified history.

6.6 CallbackData

The CallbackData contains the classes which are exchanged in callback methods.

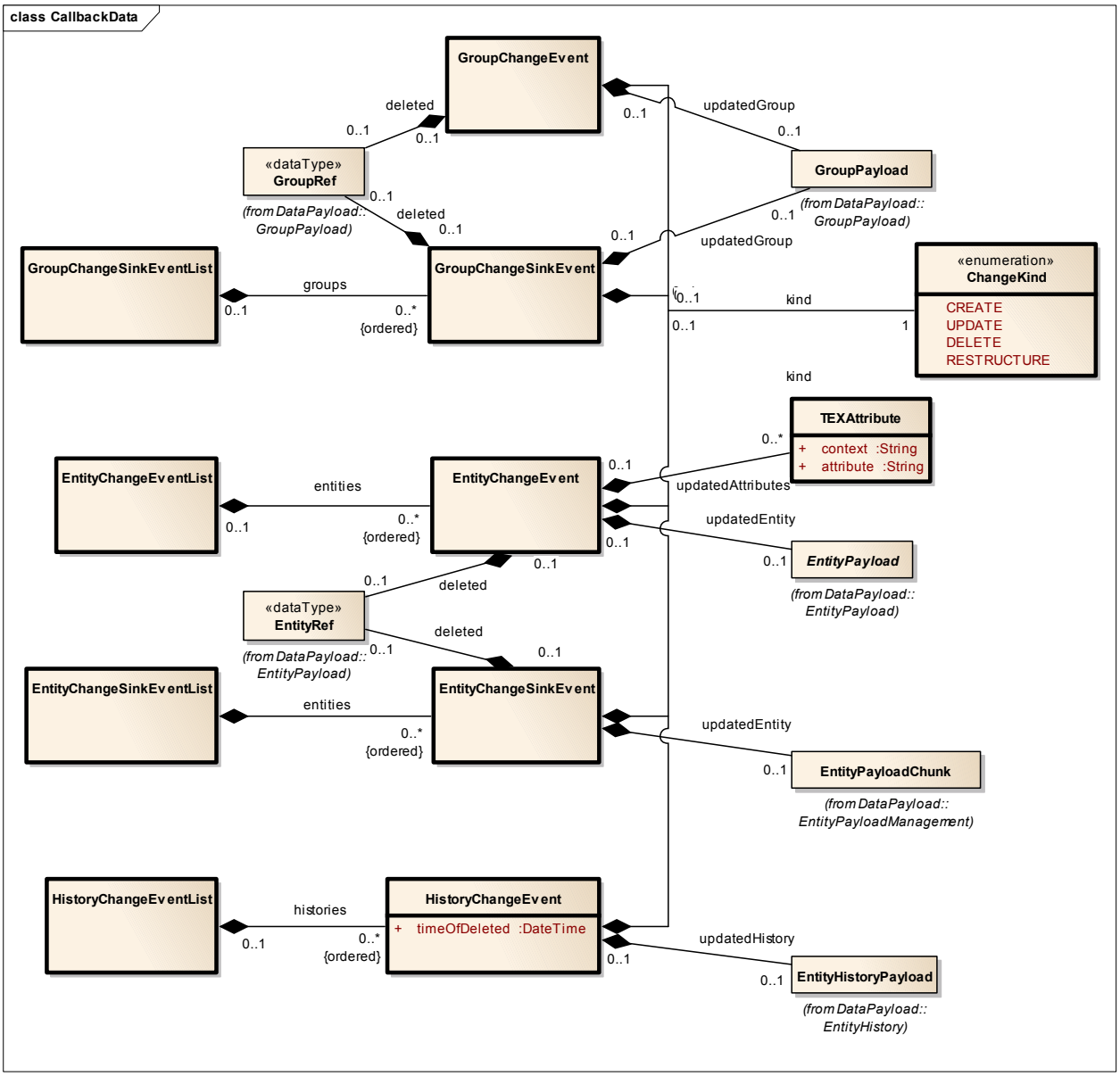


Figure 6.8 - CallbackData (Class diagram)

6.6.1 ChangeKind (Enumeration)

The enumeration of the types of entity, group or history changes.

Connections

Connector	Source	Target	Notes
Aggregation kind Source -> Destination	ChangeKind	EntityChangeSinkEvent	The operation performed by the event.
Aggregation kind	ChangeKind	GroupChangeEvent	The operation performed by the

Source -> Destination			event.
Aggregation kind Source -> Destination	ChangeKind	GroupChangeSinkEvent	The operation performed by the event.
Aggregation kind Source -> Destination	ChangeKind	HistoryChangeEvent	The operation performed by the event.
Aggregation kind Source -> Destination	ChangeKind	EntityChangeEvent	The operation performed by the event.

Attributes

Attribute	Notes	Default
CREATE	Entity/group/history creation.	
UPDATE	Entity/group/history update.	
DELETE	Entity/group/history deletion.	
RESTRUCTURE	Update of the inner structure of a CompositeEntity.	

6.6.2 EntityChangeEvent (Class)

A creation, modification or deletion event of an entity

Connections

Connector	Source	Target	Notes
Aggregation deleted Source -> Destination	EntityRef	EntityChangeEvent	The deleted entity.
Aggregation entities Source -> Destination	EntityChangeEvent ordered	EntityChangeEventList	The list of events.
Aggregation kind Source -> Destination	ChangeKind	EntityChangeEvent	The operation performed by the event.
Aggregation updatedAttributes Source -> Destination	TEXAttribute	EntityChangeEvent	Liste of the attributes that changed (when changeKind is UPDATE).
Aggregation	EntityPayload	EntityChangeEvent	The created or modified entity.

updatedEntity Source -> Destination			

6.6.3 EntityChangeEventList (Class)

List of events of creation, modification and/or deletion of entities.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	EntityChangeListener	EntityChangeEventList	
Aggregation entities Source -> Destination	EntityChangeEvent ordered	EntityChangeEventList	The list of events.

6.6.4 EntityChangeSinkEvent (Class)

An event of creations, modifications or deletions of entities used in DataSink.

Connections

Connector	Source	Target	Notes
Aggregation deleted Source -> Destination	EntityRef	EntityChangeSinkEvent	The deleted entity.
Aggregation entities Source -> Destination	EntityChangeSinkEvent ordered	EntityChangeSinkEvent List	The list of events
Aggregation kind Source -> Destination	ChangeKind	EntityChangeSinkEvent	The operation performed by the event.
Aggregation updatedEntity Source -> Destination	EntityPayloadChunk	EntityChangeSinkEvent	The created or modified entity.

6.6.5 EntityChangeSinkEventList (Class)

List of events of creation, modification and/or deletion of entities used in DataSink.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	EntityChangeSinkListener	EntityChangeSinkEvent List	

Aggregation entities Source -> Destination	EntityChangeSinkEvent ordered	EntityChangeSinkEvent List	The list of events

6.6.6 GroupChangeEvent (Class)

An event of creation, modification or deletion of group.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	GroupChangeListener	GroupChangeEvent	
Aggregation deleted Source -> Destination	GroupRef	GroupChangeEvent	The deleted group.
Aggregation kind Source -> Destination	ChangeKind	GroupChangeEvent	The operation performed by the event.
Aggregation updatedGroup Source -> Destination	GroupPayload	GroupChangeEvent	The created or modified group.

6.6.7 GroupChangeSinkEvent (Class)

A creation, modification or deletion event of a group. Group modification events include adding and removing entities as well as adding and removing entity relationships.

Connections

Connector	Source	Target	Notes
Aggregation deleted Source -> Destination	GroupRef	GroupChangeSinkEvent	The deleted group.
Aggregation groups Source -> Destination	GroupChangeSinkEvent	GroupChangeSinkEventList	The list of events.
Aggregation kind Source -> Destination	ChangeKind	GroupChangeSinkEvent	The operation performed by the event.

Aggregation updatedGroup Source -> Destination	GroupPayload	GroupChangeSinkEvent	The created or modified group.

6.6.8 GroupChangeSinkEventList (Class)

List of events of creation, modification and/or deletion of groups.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	GroupChangeSinkListener	GroupChangeSinkEventList	
Aggregation groups Source -> Destination	GroupChangeSinkEvent	GroupChangeSinkEventList	The list of events.

6.6.9 HistoryChangeEvent (Class)

An event of creations, modifications or deletions of histories.

Connections

Connector	Source	Target	Notes
Aggregation histories Source -> Destination	HistoryChangeEvent ordered	HistoryChangeEventList	The list of events.
Aggregation kind Source -> Destination	ChangeKind	HistoryChangeEvent	The operation performed by the event.
Aggregation updatedHistory Source -> Destination	EntityHistoryPayload	HistoryChangeEvent	The created or modified history.

Attributes

Attribute	Notes	Default
timeOfDeleted DateTime	The time of the deleted history if this event is a deletion event.	

6.6.10 HistoryChangeEventList (Class)

List of events of creation, modification and/or deletion of history.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	HistoryChangeListener	HistoryChangeEventList	
Aggregation histories Source -> Destination	HistoryChangeEvent ordered	HistoryChangeEventList	The list of events.

6.6.11 TEXAttribute (Class)

One of the attributes used package-wide in DataPayload.

Connections

Connector	Source	Target	Notes
Aggregation updatedAttributes Source -> Destination	TEXAttribute	EntityChangeEvent	Liste of the attributes that changed (when changeKind is UPDATE).

Attributes

Attribute	Notes	Default
context String	Name of the context class of the attribute.	
attribute String	Name of the attribute.	

6.7 EntityPayloadManagement

EntityPayloadManagement is the package of the classes needed to define the changes to the EntiotyPayload.

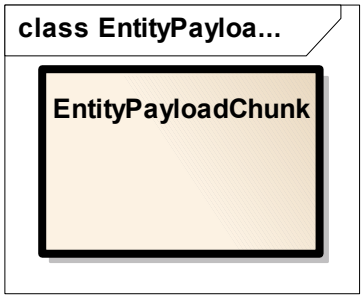


Figure 6.9 - EntityPayloadManagement (Class diagram)

6.7.1 EntityPayloadChunk (Class)

Fragment of EntityPayload. This class is further defined by the PSM used.

For example, this could be implemented as a list of key/Value pairs.

Connections

Connector	Source	Target	Notes
Aggregation updatedEntity Source -> Destination	EntityPayloadChunk	EntityChangeSinkEvent	The created or modified entity.

7 Data Interface Platform-Independent Model

The DataInterface package define the interfaces provided or requested by a TACSIT system. It uses the DataPayload package for the content of the methods of its interfaces (payload).

It also uses the Controller package from TCI.

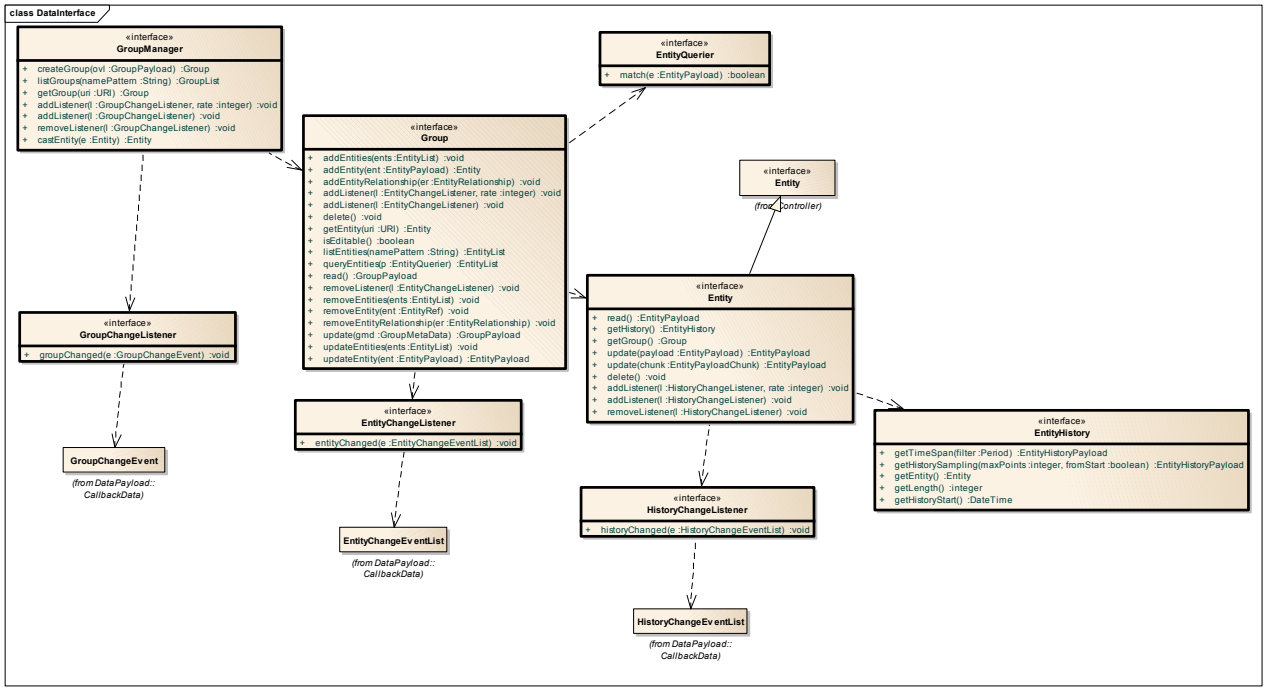


Figure 7.1 - DataInterface (Class diagram)

7.1 GroupManager (Interface)

This interface allows to create and list groups. It is the entry point of the API.

It must be noted that groups may be limited in number.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	GroupManager	GroupChangeListener	
Dependency Source -> Destination	GroupManager	Group	

Operations

Method	Parameters	Info	Notes
createGroup() Group	GroupPayload [in] ovl	throws	Create a group from the

		GroupCreationNotAllowed throws GroupAlreadyKnown, TooManyGroups, GroupCreationNotAllowed	given payload. The implementation shall return the GroupAlreadyKnown exception if the name is already used. The implementation may choose to limit the creation of groups existing in the system at the same time by returning either the TooManyGroups exception (when the number of groups is limited by the implementation) or the GroupCreationNotAllowed exception (when the creation of new groups is not allowed by the implementation).
listGroups() GroupList	String [in] namePattern		List the references of the already defined groups filtered by their name (namePattern is a regexp).
getGroup() Group	URI [in] uri	throws UnknownURI	Get the interface to a group from its URI and return the UnknownURI exception if the entity does not exist (anymore).
addListener() void	GroupChangeListener [in] l integer [in] rate	throws InadequateRate, TooManyListeners, NotImplemented	Listen to the group creation, deletion, and update events at a given rate. This method may return the InadequateRate exception (if the implementation deems this parameter as unsuitable) or the TooManyListeners exception (if the implementation limit the number of listeners) or the NotImplemented exception if this capability is not available.
addListener() void	GroupChangeListener [in] l	throws	Listen to the group

		TooManyListerners	creation, deletion, and update events. This method may return the TooManyListeners exception (if the implementation limit the number of listeners).
removeListener() void	GroupChangeListener [in] l	throws UnknownListerner	Remove a previously set listener and return the UnknownListener exception if the listener does not exist (any more).
castEntity() Entity	Entity [in] e	throws UnknownEntity	Translate an Entity from the package Controller (from standard TCI) to the local Entity (from TEX). If not possible, the exception EntityUknown is thrown.

7.2 Group (Interface)

This interface accesses to a group.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	GroupManager	Group	
Dependency Source -> Destination	Group	Entity	
Dependency Source -> Destination	Group	EntityChangeListener	
Dependency Source -> Destination	Group	EntityQuerier	

Operations

Method	Parameters	Info	Notes
addEntities() void	EntityList [in] ents	throws NotImplemented	Grouped creation of entities. The implementation may return a NotImplemented

			exception.
addEntity() Entity	EntityPayload [in] ent	throws NotImplemented	Add an entity to the group and returns the possibly modified entity with its URI. The implementation may return a NotImplemented exception.
addEntityRelationship() void	EntityRelationship [in] er		Adds an entity relationship (a pair-wise relation between two of the group's entities) to the group. The implementation may return a NotImplemented exception.
addListener() void	EntityChangeListener [in] l integer [in] rate	throws InadequateRate, TooManyListerner, NotImplemented	Listen to the entity creation, deletion, and update events in the group at a given rate. This method may return the InadequateRate exception (if the implementation deems this parameter as unsuitable) or the TooManyListeners exception (if the implementation limit the number of listeners) or the NotImplemented exception if this capability is not available.
addListener() void	EntityChangeListener [in] l	throws TooManyListerner	Listen to the entity creation, deletion, and update events in the group. This method may return the TooManyListeners exception (if the implementation limit the number of listeners).
delete() void		throws NotImplemented	Delete the group. The implementation may return a NotImplemented exception.
getEntity() Entity	URI [in] uri	throws UnknownURI	Get the interface of an

			entity from its URI and return the UnknownURI exception if the entity does not exist (anymore).
isEditable() boolean			Returns true if the group may be edited according to the TACSIT system.
listEntities() EntityList	String [in] namePattern		List the references to the entities of the group after filtering them on their name (namePattern is a regexp).
queryEntities() EntityList	EntityQuerier [in] p		List the references to the entities of the group that match the predicate in argument.
read() GroupPayload			Get the payload of the group.
removeListener() void	EntityChangeListener [in] l	throws UnknownListener	Remove a previously set listener and return the UnknownListener exception if the listener does not exist (any more).
removeEntities() void	EntityList [in] ents	throws NotImplemented	Grouped retrieval of entities. The implementation may return a NotImplemented exception.
removeEntity() void	EntityRef [in] ent	throws UnknownEntity, NotImplemented	Remove an entity in the group and return the UnknownEntity exception if the entity does not exist (anymore). The implementation may return a NotImplemented exception.
removeEntityRelationship( ) void	EntityRelationship [in] er		Removes an entity relationship (a pair-wise relation between two of the group's entities) to the group. The implementation may return a NotImplemented exception or the Group has no such entity relationship the UnknownEntityRelations

			hip exception
update() GroupPayload	GroupMetaData [in] gmd	throws NotImplemented	Update the meta data of the group and return the new group. The implementation may return a NotImplemented exception.
updateEntities() void	EntityList [in] ents	throws NotImplemented	Grouped modification of entities. The implementation may return a NotImplemented exception.
updateEntity() EntityPayload	EntityPayload [in] ent	throws NotImplemented	Modify an entity in the group (based on its URI) and returns the possibly modified entity. The implementation may return a NotImplemented exception.

7.3 Entity (Interface)

This interface accesses to an entity.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Entity	Entity	
Dependency Source -> Destination	Group	Entity	
Dependency Source -> Destination	Entity	HistoryChangeListener	
Dependency Source -> Destination	Entity	EntityHistory	

Operations

Method	Parameters	Info	Notes
read() EntityPayload			Get the payload of the entity.
getHistory() EntityHistory			Get the interface to the

			history of the entity.
getGroup() Group			Get back the matching Group interface.
update() EntityPayload	EntityPayload [in] payload	throws NotImplemented	Update the entity and return the new payload. The implementation may return a NotImplemented exception.
update() EntityPayload	EntityPayloadChunk [in] chunk	throws NotImplemented	Update the entity with a subset of the data defined in an EntityPayload and return the new payload. The implementation may return a InconsistentChunk exception. The implementation may return a NotImplemented exception.
delete() void		throws NotImplemented	Delete the entity. The implementation may return a NotImplemented exception.
addListener() void	HistoryChangeListener [in] l integer [in] rate	throws InadequateRate, TooManyListerners, NotImplemented	Listen to the history creation, deletion, and update events for the entity at a given rate. This method may return the InadequateRate exception (if the implementation deems this parameter as unsuitable) or the TooManyListeners exception (if the implementation limit the number of listeners) or the NotImplemented exception if this capability is not available.
addListener() void	HistoryChangeListener [in] l	throws TooManyListerners	Listen to the history creation, deletion, and update events for the entity. This method may return the TooManyListeners exception (if the

			implementation limit the number of listeners).
removeListener() void	HistoryChangeListener [in] l	throws UnknownListener	Remove a previously set listener and return the UnknownListener exception if the listener does not exist (any more).

7.4 EntityHistory (Interface)

This interface access to the history of an entity.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	Entity	EntityHistory	

Operations

Method	Parameters	Info	Notes
getTimeSpan() EntityHistoryPayload	Period [in] filter		Get the payload of the history filtered on the date (no history before this date will be returned).
getHistorySampling() EntityHistoryPayload	integer [in] maxPoints boolean [in] fromStart		Get the payload of the history sampled with a 'maxPoints' maximum number of points. If 'fromStart' is false, the les than 'maxPoints' most recent data are returned. Otherwise, an equal distribution of 'maxPoints' data taken among the known points is returned.
getEntity() Entity			Get back the matching Entity interface.
getLength() integer			Get the number of data in the history.
getHistoryStart() DateTime			Get the date of the first data in the history.

7.5 GroupChangeListener (Interface)

Interface invoked as a callback when a group is created, modified or deleted.

Connections

Connector	Source	Target	Notes

Dependency Source -> Destination	GroupManager	GroupChangeListener	
Dependency Source -> Destination	GroupChangeListener	GroupChangeEvent	

Operations

Method	Parameters	Info	Notes
groupChanged() void	GroupChangeEvent [in] e	throws StopListening	This method is called for each creation, modification and deletion of group. The implementation of this callback may return the exception StopListening as soon it wants to kill the listeners.

7.6 EntityChangeListener (Interface)

Interface invoked as a callback when entities are created, modified or deleted within a group.

The implementation of this callback may return the exception StopListening as soon it wants to kill the listeners.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	Group	EntityChangeListener	
Dependency Source -> Destination	EntityChangeListener	EntityChangeEventList	

Operations

Method	Parameters	Info	Notes
entityChanged() void	EntityChangeEventList [in] e	throws StopListening	Interface invoked as a callback when a bulk of entity creation, modification or deletion happens.

7.7 HistoryChangeListener (Interface)

Interface invoked as a callback when histories are created, modified or deleted for an entity.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	Entity	HistoryChangeListener	
Dependency Source -> Destination	HistoryChangeListener	HistoryChangeEventList	

Operations

Method	Parameters	Info	Notes
historyChanged() void	HistoryChangeEventList [in] e	throws StopListening	Interface invoked as a callback when a bulk of history creation, modification or deletion happens. The implementation of this callback may return the exception StopListening as soon it wants to kill the listeners.

7.8 EntityQuerier (Interface)

Predicate function for the query service.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	Group	EntityQuerier	

Operations

Method	Parameters	Info	Notes
match() boolean	EntityPayload [in] e		Predicate function returning true if e matches the predicate.

8 Data Sink Interface Platform-Independent Model

The DataSink package contains the interfaces that a feeding data server must implement. It uses the DataPayload package for the content of the methods of its interfaces (payload).

It also uses the Query package from TCI.

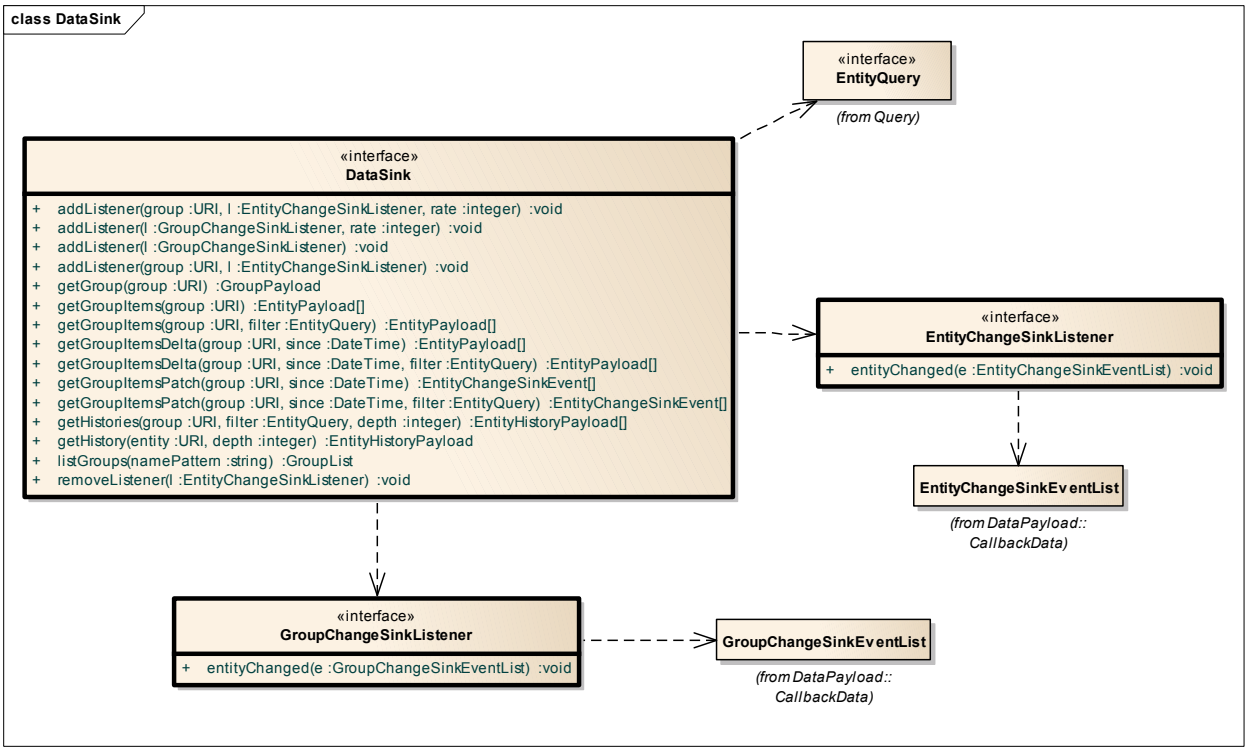


Figure 8.1 - DataSink (Class diagram)

8.1 DataSink (Interface)

This interface is requested by a TACSIT system to get data from a business server.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	DataSink	GroupChangeSinkListener	
Dependency Source -> Destination	DataSink	EntityQuery	
Dependency Source -> Destination	DataSink	EntityChangeSinkListener	

Operations

Method	Parameters	Info	Notes
addListener() void	URI [in] group EntityChangeSinkListener [in] l integer [in] rate	throws UnknownURI, IsNotAGroup, InadequateRate, TooManyListerners	Listen to the entity creation, deletion, and update events in a group. The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore). It may return the InadequateRate exception (if the implementation deems this parameter as unsuitable) or the TooManyListeners exception (if the implementation limit the number of listeners).
addListener() void	GroupChangeSinkListener [in] l integer [in] rate	throws UnknownURI, IsNotAGroup, InadequateRate, TooManyListerners	Listen to the create, update and delete events for a group. It may return the InadequateRate exception (if the implementation deems this parameter as unsuitable) or the TooManyListeners exception (if the implementation limit the number of listeners).
addListener() void	GroupChangeSinkListener [in] l	throws UnknownURI, IsNotAGroup, InadequateRate, TooManyListerners	Listen to the create, update and delete events for a group. It may return the TooManyListeners exception (if the implementation limit the number of listeners).
addListener() void	URI [in] group EntityChangeSinkListener [in] l	throws UnknownURI, IsNotAGroup, TooManyListerners	Listen to the entity creation, deletion, and update events in a group. The method shall return

			the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore). It may return the TooManyListeners exception (if the implementation limit the number of listeners).
getGroup() GroupPayload	URI [in] group	throws UnknownURI, IsNotAGroup	Invoked by the TACSIT system to get the definition of a group from its URI. The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore).
getGroupItems() EntityPayload	URI [in] group	throws UnknownURI, IsNotAGroup	Invoked by the TACSIT system to get the list of the entities of a group given by an URI. The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore).
getGroupItems() EntityPayload	URI [in] group EntityQuery [in] filter	throws UnknownURI, IsNotAGroup	Invoked by the TACSIT system to get the filtered list of the entities of a group given by an URI. The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore).
getGroupItemsDelta() EntityPayload	URI [in] group DateTime [in] since	throws UnknownURI, IsNotAGroup	Invoked by the TACSIT system to get the list of the entities of a group given by an URI. The returned entities are only those that have been modified since the given date.

			The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore).
getGroupItemsDelta() EntityPayload	URI [in] group DateTime [in] since EntityQuery [in] filter	throws UnknownURI, IsNotAGroup	Invoked by the TACSIT system to get the filtered list of the entities of a group given by an URI. The returned entities are only those that have been modified since the given date. The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore).
getGroupItemsPatch() EntityChangeSinkEvent	URI [in] group DateTime [in] since	throws UnknownURI, IsNotAGroup	Invoked by the TACSIT system to get the list of the entity deltas of a group given by an URI. The returned deltas are only those that have been modified since the given date. The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore).
getGroupItemsPatch() EntityChangeSinkEvent	URI [in] group DateTime [in] since EntityQuery [in] filter	throws UnknownURI, IsNotAGroup	Invoked by the TACSIT system to get the list of the entity deltas of a group given by an URI. The returned deltas are only those that have been modified since the given date. The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore).
getHistories()	URI [in] group	throws UnknownURI,	Invoked by the TACSIT

EntityHistoryPayload	EntityQuery [in] filter integer [in] depth	IsNotAGroup	system to get the histories of the entities of a group and which match a given Query within a given depth. The method shall return the UnknownURI or IsNotAGroup exceptions if the URI does not identify a group (anymore).
getHistory() EntityHistoryPayload	URI [in] entity integer [in] depth	throws UnknownURI, IsNotAnEntity	Invoked by the TACSIT system to get the history of an entity with a given depth. The method shall return the UnknownURI or IsNotAnEntity exceptions if the URI does not identify an entity (anymore).
listGroups() GroupList	string [in] namePattern		Invoked by the TACSIT system to get the list of the known groups filtered by their name (namePattern is a regexp)..
removeListener() void	EntityChangeSinkListener [in] l	throws UnknownListerner	This interface accesses to a group and return the UnknownListener exception if the listener does not exist (any more).

8.2 EntityChangeSinkListener (Interface)

Interface invoked as a callback when entities are created, modified or deleted within a group.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	DataSink	EntityChangeSinkListener	
Dependency Source -> Destination	EntityChangeSinkListener	EntityChangeSinkEventList	

Operations

Method	Parameters	Info	Notes
entityChanged() void	EntityChangeSinkEventList [in] e	throws StopListening	Interface invoked as a callback when a bulk of entity creation, modification or deletion happens. The implementation of this callback may return the exception StopListening as soon it wants to kill the listeners.

8.3 GroupChangeSinkListener (Interface)

Interface invoked as a callback when groups are created, modified or deleted.

Connections

Connector	Source	Target	Notes
Dependency Source -> Destination	DataSink	GroupChangeSinkListener	
Dependency Source -> Destination	GroupChangeSinkListener	GroupChangeSinkEventList	

Operations

Method	Parameters	Info	Notes
entityChanged() void	GroupChangeSinkEventList [in] e		Interface invoked as a callback when a bulk of group modification (includes add and remove entities as well as creation and deletion of entity relationships), creation and deletion happens . The implementation of this callback may return the exception StopListening as soon it wants to kill the listeners.

9 Data Payload Platform-Specific Models

9.1 Payload Media Types

Several of the Data Interface PSM or Data Sink PSM are generic of the Data Payload PSM they use. For that purpose, they need a codification of the known Data Payload PSM. This codification is obtained by the use of media types, also known as MIME types (see [MIME]).

Data Payload PSM supported by this specification and the matching media types (see [MIME]) are:

- application/x.tacsit+xml: See section 10.2;
- application/x.tacsit+json: reserved;
- application/x.tacsit-nvg+xml: See section 10.3;
- application/x.tacsit-nvg+json: reserved.

A version may be appended to the media type as specified by [MIME], i.e. after a semicolon. This version shall be the version number of this standard, e.g.: application/x.tacsit+xml; version=1.0.0.

9.2 XML PSM

This is a placeholder for a future PSM.

9.3 Java PSM

This is a placeholder for a future PSM.

9.4 C# PSM

The C# PSM maps the Data Payload PIM classes to C# classes using MDA code generation. The detailed rules for the code generation are as follows:

- The PIM attributes are mapped to a C# class private member attributes (camel case) and a public read-write property (Pascal case);
- A public constructor is defined for all the mandatory attributes;
- Optional attributes that cannot take a null value (enumerations and scalar data-types) are mapped to an instantiation of the System.Nullable class;
- Ordered collections in the PIM are mapped to an instantiation of the IList interface;
- Unordered collections in the PIM are mapped to an instantiation of the IEnumerable interface;
- Specialization / Generalization PIM relationships are mapped to C# class inheritance;
- The String, DateTime and URI classes are mapped to their C# built-in class library equivalents;
- The Period class is mapped to a class with a pair of DateTime attributes.

9.5 DDS PSM

The DDS PSM defines a set of IDL files for the Data Payload PIM classes. DDS provides the functionality of the Data Interface and Data Sink PIMs, therefore no specific DDS PSM is provided for these. The Data Payload PSM defines the following DDS topic types:

- EntityPayload,
- GroupPayload.

The detailed rules for the MDA code generation from the Data Payload PIM to the DDS PSM IDL are as follows:

- The PIM attributes are mapped to IDL attributes;

- Optional attributes are mapped to a union type with a single member present when the exists case attribute is true;
- Collections in the PIM are mapped to IDL sequences;
- Specialization / Generalization PIM relationships are mapped to IDL unions. Additional data classes are introduced for generalization classes that have attributes ;
- The String and URI classes are mapped to an IDL string;
- The DataTime and Period classes are mapped to unsigned long long and a struct with a pair of DateTime attributes.

9.6 NVG PSM

9.6.1 Overview and Limitations

The NATO Vector Graphic (see [NVG]) STANAG provides a simple specification for encoding battle-space information to support geospatial viewing based on an XML file exchange format. The NVG PSM maps the TEX PIM onto the NVG Data Format allowing sending and receiving TEX data as NVG-formatted data.

This PSM:

- Deals only with geodetic coordinates (latitude/longitude);
- Does not deal with annulus entities.

9.6.2 Mapping

The following table specifies the mapping between TEX PIM and the NVG 2.0 schema by showing the correspondence between on the one hand TEX classes and attributes and on the other hand NVG XML elements.

In this table, "@X" references an attribute (TEX or NVG) named X and elements standardized by NVG and actually reused are written in bold font.

The needed and referenced XSD files are provided separately.

Table 1: Mapping between TEX and NVG 2.0

TEX			NVG			Comment
GroupRef			URL			
GroupList			List of URLs			
GroupPayload			**<nvg:nvg>**			
	@id			@uri		
	@version			**@version**		
	entities			See EntityPayload		Here entities are included as is and not through references.
	metadata					The content of TEX metadata is spread along 3 NVG elements: metadata, extended data and schema
				<nvg:metadata>		The content of the metadata element shall match the GroupMetaData XSD

TEX			NVG			Comment
		@isReadOnly			@readonly	boolean
		CoordinateUnits			units	
	links				links	
				<nvg:ExtendedData schemaRef="TEX:metadata">		
		@publisher			**<nvg:SimpleData key="dcterms:publisher">**	
		@identifier			**<nvg:SimpleData key="dcterms:identifier">**	
		@securityPolicy			**<nvg:SimpleData key="dcterms:security.policy">**	
		@securityClassification			**<nvg:SimpleData key="dcterms:security.classification">**	
		@securityCategory			**<nvg:SimpleData key="dcterms:security.category">**	
		@category			**<nvg:SimpleData key="dcterms:subject.category">**	
		schemas		**<nvg:Schema schemaId="name">**		
EntityRef			URL			
EntityList			List of URLs			
EntityPayload			None			Abstract class
	@id			**@uri**		
	@label			**@label**		
	@info			**<nvg:textInfo>**		
	@href			**@href**		
	@timeSpan			**<nvg:TimeSpan>**		See [NVG] for the actual format.
	@timeStamp			**<nvg:TimeStamp>**		See [NVG] for the actual format.
	metaData			**<nvg:ExtendedData schemaRef="TEX:metadata">**		
		@publisher			**<nvg:SimpleData key="dcterms:publisher">**	
		@identifier			**<nvg:SimpleData key="dcterms:identifier">**	
		@securityPolicy			**<nvg:SimpleData key="dcterms:security.policy">**	
		@securityClassification			**<nvg:SimpleData key="dcterms:security.classification">**	
		@securityCategory			**<nvg:SimpleData key="dcterms:security.category">**	
	@reportType			@reportType		Stringified enumerate
ShapedEntity			None			Abstract class
	See EntityPayload					
	@directionOfMovement			@directionOfMovement		
	@isHeadingRelative			@isHeadingRelative		
	@climbAngle			@climbAngle		
	@speed			@speed		
	@categorization			**@symbol**		Value is SymbolSet + ":" + SymbolId as specified in the categorization class.
	@categorization.modifiers			**@modifiers**		For APP-6B
	@categorizationIn3D			@3Dsymbol		Content of @extrudeHeight
	@legType			@leg-type		Stringified enumerate.

TEX			NVG			Comment
	extendedData			**<nvg:ExtendedData>**		
		schema			**<nvg:Section schemaRef="***schema***">**	
		@key				**<nvg:SimpleData key="***key***">**
		@value				*value*
AggregateEntity			None			Abstract class
	See EntityPayload					
	groupedEntities			See EntityPayload		
Bearing			**<nvg:content-item>**			
	See ShapedEntity					
	@azimuth			**<nvg:ExtendedData schemaRef="**TEX:Bearing**">** **<nvg:SimpleData key="**azimuth**">** *angle* **</nvg:SimpleData> </nvg:ExtendedData>**		
	point					Only GeodeticPosition
		@latitude		**@x**		
		@longitude		**@y**		
		@altitude		@z		
AmbiguousBearing			**<nvg:content-item>**			
	See ShapedEntity					
	@bearingA @bearingB			**<nvg:ExtendedData schemaRef="**TEX:AmbiguousBearing**">** **<nvg:SimpleData key="**bearingA**">** *angle* **</nvg:SimpleData>** **<nvg:SimpleData key="**bearingB**">** *angle* **</nvg:SimpleData></nvg:ExtendedData>**		
	point					Only GeodeticPosition
		@latitude		**@x**		
		@longitude		**@y**		
		@altitude		@z		
Text			**<nvg:text>**			
	See ShapedEntity					
	@content			**<nvg:content>**		
	@rotation			**@rotation**		
	point					Only GeodeticPosition
		@latitude		**@x**		
		@longitude		**@y**		
		@altitude		@z		
Point			**<nvg:point>**			

TEX			NVG			Comment
	See ShapedEntity					
	center					Only GeodeticPosition
		@latitude		**@x**		
		@longitude		**@y**		
		@altitude		**@z**		
Multipoint			**<nvg:multipoint>**			
	See ShapedEntity					
	@rotation			**@rotation**		
	points			**@points**	See [NVG] for the formatting of this attribute (list of (Longitude,Latitude) couples). Only GeodeticPosition and no altitude	
Circle			**<nvg:circle>**			
	See ShapedEntity					
	@radius			**@r**		
	center					Only GeodeticPosition
		@latitude		**@cx**		
		@longitude		**@cy**		
		@altitude		**@minaltitude** **@maxaltitude**		
Ellipse			**<nvg:ellipse>**			
	See ShapedEntity					
	@NSSemiAxis			**@rx**		
	@EWSemiAxis			**@ry**		
	@rotation			**@rotation**		
	center					Only GeodeticPosition
		@latitude		**@cx**		
		@longitude		**@cy**		
		@altitude		**@minaltitude** **@maxaltitude**		
Rectangle			**<nvg:rect>**			
	See ShapedEntity					
	@NSHalfDistance			**@rx**		
	@EWHalfDistance			**@ry**		
	@rotation			**@rotation**		
	center					Only GeodeticPosition
		@latitude		**@cx**		
		@longitude		**@cy**		
		@altitude		**@minaltitude** **@maxaltitude**		

TEX			NVG			Comment
Polyline			**<nvg:polyline>**			
	See ShapedEntity					
	points			**@points**	See [NVG] for the formatting of this attribute (list of (Longitude,Latitude) couples). Only GeodeticPosition and no altitude	
Arrow			**<nvg:arrow>**			
	See ShapedEntity					
	@width			**@width**		
	points			**@points**	See [NVG] for the formatting of this attribute (list of (Longitude,Latitude) couples). Only GeodeticPosition and no altitude	
Corridor			**<nvg:corridor>**			
	See ShapedEntity					
	@width			**@width**		
	points			**@points**	See [NVG] for the formatting of this attribute (list of (Longitude,Latitude) couples). Only GeodeticPosition and no altitude	
Orbit			**<nvg:orbit>**			
	See ShapedEntity					
	@width			**@width**		
	pointOne pointTwo			**@points**	See [NVG] for the formatting of this attribute (list of 2 (Longitude,Latitude) couples). Only GeodeticPosition and no altitude	
Polygon			**<nvg:polygon>**			
	See ShapedEntity					
	points			**@points**	See [NVG] for the formatting of this attribute (list of (Longitude,Latitude) couples). Only GeodeticPosition and no altitude	
Arc			**<nvg:arc>**			
	See ShapedEntity					
	@NSSemiAxis			**@rx**		
	@EWSemiAxis			**@ry**		
	@startAngle			**@startangle**		
	@endAngle			**@endangle**		
	@rotation			**@rotation**		
	center					Only GeodeticPosition
		@latitude		**@cx**		
		@longitude		**@cy**		
		@altitude		**@minaltitude** **@maxaltitude**		

TEX			NVG			Comment
Arcband			**<nvg:arcband>**			
	See ShapedEntity					
	@minRadius			**@minr**		
	@maxRadius			**@maxr**		
	@startAngle			**@startangle**		
	@endAngle			**@endangle**		
	center					Only GeodeticPosition
		@latitude		**@cx**		
		@longitude		**@cy**		
		@altitude		**@minaltitude** **@maxaltitude**		
StickyNote			**<nvg:content-item>**			
	See ShapedEntity					
	@text			**<nvg:ExtendedData schemaRef="**TEX:StickyNote**">** **<nvg:SimpleData key="dcterms:title">** *text* **</nvg:SimpleData> </nvg:ExtendedData>**		
	@font			@font		
	@textColor			@textcolor		
	@backgroundColor			@backgroundcolor		
	@borderStyle			@borderstyle		
	@offsetX			@offsetx		
	@offsetY			@offsety		
	center					Only GeodeticPosition
		@latitude		@cx		
		@longitude		@cy		
Annulus			No mapping			
FreeShapedEntity			**<nvg:content-item>**			
	See ShapedEntity					
	@svgDefinition			**<nvg:ExtendedData schemaRef="**TEX:FreeShapedEntity**">** **<nvg:SimpleData key="**svgDefinition**">** *SVG content* **</nvg:SimpleData>** **<nvg:SimpleData key="**bearingB**">** *angle* **</nvg:SimpleData></nvg:ExtendedData>** Only GeodeticPosition		
	point					
		@latitude		**@x**		
		@longitude		**@y**		
		@altitude		@z		

TEX			NVG			Comment
EntityHistoryPayload			<nvgtex:history>			See EntityHistoryPayload XSD
	reference			@id		See EntityRef
				See EntityPayload		
EntityHistoryList			<nvgtex:histories>			See EntityHistoryPayload XSD
	histories			See EntityHistoryPayload		
GroupChangeEvent			<nvgtex:GroupChangeEvent>			See GroupChangeEvent XSD
	@kind			@kind		
	deleted			@deleted		See GroupRef
	updatedGroup			See GroupPayload		
GroupChangeSinkEventList			<nvgtex:GroupChangeSinkEventList>			See GroupChangeSinkEvent XSD
				See GroupChangeSinkEvent		
GroupChangeSinkEvent			<nvgtex:GroupChangeSinkEvent>			See GroupChangeSinkEvent_XSD
	@kind			@kind		
	deleted			@deleted		See GroupRef
				See GroupPayload		
EntityChangeEventList			<nvgtex:EntityChangeEventList>			See EntityChangeEvent_XSD
				See EntityChangeEvent		
EntityChangeEvent			<nvgtex:EntityChangeEvent>			See EntityChangeEvent_XSD
	@kind			@kind		
	updatedAttributes			<nvgtex:updatedAttributes>		
		@context			@context	
		@attribute			@attribute	
	deleted			@deleted		See EntityRef
				See EntityPayload		
EntityChangeSinkEventList			<nvgtex:EntityChangeSinkEventList>			See EntityChangeSinkEvent XSD
				See EntityChangeSinkEvent		
EntityChangeSinkEvent			<nvgtex:EntityChangeSinkEvent>			See EntityChangeSinkEvent_XSD
	@kind			@kind		
	deleted			@deleted		See EntityRef
				See EntityPayload		
HistoryChangeEventList			<nvgtex:HistoryChangeEventList>			See HistoryChangeEventList_XSD
				See HistoryChangeEvent		

TEX			NVG			Comment
HistoryChangeEvent			<nvgtex:HistoryChangeEvent>			See HistoryChangeEvent_XSD
	@kind			@kind		
	updatedEntity			See EntityPayload		
EntityPayloadChunk			same as EntityPayload			

9.7 NVGjs PSM

This is a placeholder for a future PSM based on current work by NATO on a JSON version of NVG.

This page intentionally left blank.

10 Data Interface Platform-Specific Models

10.1 Java PSM

This is a placeholder for a future PSM.

10.2 C# PSM

The C# PSM maps the Data Interface PIM classes and interfaces to C# classes and interfaces using MDA code generation. The detailed rules for the code generation are as follows:

- PIM classes are mapped to C# classes;
- PIM interfaces are mapped to C# interfaces;
- Listener interfaces are mapped to the C# event mechanism; rate limited or unlimited events are accessed through a factory method;
- The PIM attributes are mapped to a C# class private member attributes (camel case) and a public read-write property (Pascal case);
- A public constructor is defined for all the mandatory attributes;
- Optional attributes that cannot take a null value (enumerations and scalar data-types) are mapped to an instantiation of the System.Nullable class;
- Ordered collections in the PIM are mapped to an instantiation of the IList interface;
- Unordered collections in the PIM are mapped to an instantiation of the IEnumerable interface;
- Specialization / Generalization PIM relationships are mapped to C# class inheritance.

10.3 DDS PSM

The DDS PSM for the Data Interface is implicitly provided by DDS DCPS using the topic types defined by the Data Payload PSM. DDS partitions are used to separate and identity different TACSIT and client instances.

10.4 TypeScript PSM

10.4.1 Overview and Limitations

TypeScript (See [TS]) is a strict syntactical superset of ECMAScript 2015 (See [ECMAScript]), and adds optional static typing to the language. TypeScript was chosen as target of PSM since it is free and open-source and supports the concepts of interface, namespace as well as definition files that can contain type information. This allows providing this PSM as definition files.

The mapping is quite direct and consists in transforming interface to interface and method to method. The datatypes are transformed in their TypeScript equivalents: integer to number, string to string and boolean to boolean. All Data Payload classes are mapped to only one class: DataPayload. This class allows the PSM to remain generic from the payload PSM by providing two attributes:

- media-type is a string that contains the media type of the PSM as specified in Section 10.1;
- payload is an Object that contains the data itself and that needs to be casted accordingly to the media type and to the awaited type.

The TypeScript PSM is not foreseen to use other Data Payload PSM than the XML and JSON-based ones.

10.4.2 Mapping

The PSM is provided separately as two files:

- tacsit-tex-datapayload.d.ts declares the module tacsit as well as the DataPayload class;

- tacsit-tex-datainterface.d.ts declares the interfaces as specified by the Data Interface PIM.

These files depends upon TCI: See Annex B.

11 Data Sink Platform-Specific Models

11.1 C# PSM

The C# PSM maps the Data Sink PIM classes and interfaces to C# classes and interfaces using MDA code generation. The detailed rules for the code generation are as follows:

- PIM classes are mapped to C# classes;
- PIM interfaces are mapped to C# interfaces;
- Listener interfaces are mapped to the C# event mechanism; rate limited or unlimited and group specific events are accessed through a factory method;
- The PIM attributes are mapped to a C# class private member attributes (camel case) and a public read-write property (Pascal case);
- A public constructor is defined for all the mandatory attributes;
- Optional attributes that cannot take a null value (enumerations and scalar data-types) are mapped to an instantiation of the System.Nullable class;
- Ordered collections in the PIM are mapped to an instantiation of the IList interface;
- Unordered collections in the PIM are mapped to an instantiation of the IEnumerable interface;
- Specialization / Generalization PIM relationships are mapped to C# class inheritance.

11.2 DDS PSM

The DDS PSM for the Data Interface is implicitly provided by DDS DCPS using the topic types defined by the Data Payload PSM. DDS partitions are used to separate and identity different TACSIT and client instances.

11.3 HTTP PSM

11.3.1 Overview and Limitations

Any instantiation of a Data Sink can be represented as Hypertext Transfer Protocol (see [HTTP]) resources. These resources are referenced by a base URL that depends on the implementation. See [RFC3986], section 5 for more details. This base URL will be denoted as baseURL throughout this section.

The resources considered by this mapping are:

- {baseURL}/groups for the TACSIT groups;
- {baseURL}/groups/{id} for a specific group;
- {baseURL}/entities/{id} for a specific entity;
- {baseURL}/listeners for the Data Listeners.

As explained in the design rationale, this PSM does not address data modeling of the payload since this is coped with by the “Data Payload PSM” sections. Yet, The HTTP PSM is not foreseen to use other Data Payload PSM than the XML and JSON-based ones.

This PSM does not map the listener methods because this exchange pattern do not fit with the simple HTTP style. Future version of this specification may consider the use of server-sent events over HTTP (See [SSE]).

11.3.2 General Conventions and Considerations

11.3.2.1 Response Codes

Any HTTP operation (see [HTTP], section 9) on a given resource that is not implemented must return an HTTP response with a status code of 405 (Method Not Allowed).

Other HTTP status codes may be added by security mechanisms or other extensions.

11.3.2.2 Content Compression

For improved performance it is recommended that the server support client requests for GZIP compression. Clients will request compression by setting the “Accept-Encoding” HTTP header to “gzip.” The server should honor this request for all documents, so that devices may benefit from the reduced bandwidth needs and improved battery life when requesting compressed content.

11.3.2.3 Media Types

See Section 10.1 for the payload media types supported by this specification.

This PSM nevertheless uses another media type:

- application/x.tacsit-link: a (list of) URL.

A version may also be appended to this media type, e.g.: application/x.tacsit-link; version=1.0.0.

11.3.3 Mapping

11.3.3.1 Mapping of Entity Queries

This mapping from an EntityQuery (see [TCI]) to a string is needed by the following interfaces.

It is implementation-dependent.

11.3.3.2 URL: {baseURL}/groups – Verb: GET

This resource maps the following method:

- DataSink:listGroups.

The “Accept” request header must be set to application/x.tacsit-link.

The “If-Modified-Since” and “If-Unmodified-Since” request headers may be set.

The request may have the following query field:

- namePattern=”string”: the namePattern argument of the getGroups method.

The “Content-Type” response header must be application/x.tacsit-link.

The “Last-Modified” request header must be set.

The payload of the response is a list of URL representing groups: BaseURL/Groups/{id} where id is the identifier of a group. These URL may subsequently be invoked to get more information for given groups.

The return code must be:

- 200 (OK): successful request;
- 204 (Empty): successful request, the resulting list is empty;

- 304 (Not Modified): the “If-Modified-Since” request header was used and the resulting list did not change meanwhile;
- 400 (Bad Request): namePattern cannot be parsed;
- 412 (Precondition Failed): the “If-Unmodified-Since” request header was used and the resulting list did change meanwhile;
- 415 (Unsupported Media Type): the “Accept” request header is wrongly set;
- 5xx range: implementation-specific server problem.

11.3.3.3 URL: {baseURL}/groups/{id} – Verb: Get

In the URL, id is the identifier of a known group.

This resource maps the following methods:

- DataSink:getGroup(URI),
- DataSink:getGroupItems(URI),
- DataSink:getGroupItemsDelta(URI, DateTime),
- DataSink:getGroupItems(URI, EntityQuery),
- DataSink:getGroupItemsPatch(URI, DateTime),
- DataSink:getGroupItemsDelta(URI, DateTime, EntityQuery),
- DataSink:getGroupItemsPatch(URI, DateTime, EntityQuery),
- DataSink:getHistories(URI, EntityQuery,int).

The “Accept” request header must be set to any payload media types (i.e. not application/x.tacsit-link).

The “If-Modified-Since” and “If-Unmodified-Since” request headers may be set.

The request may have the following query field:

- since=”date”: the “since” argument of the methods taking a DateTime argument (“date” is an ISO date);
- entityQuery=”string”: the “filter” argument of the methods taking an EntityQuery argument; the mapping of such an EntityQuery to a string is specified in section 12.1.3.1;
- return=”content|patch|delta|history”: if “content”, the method is “getGroupItems”, if “patch”, the method is “getGroupItemsPatch”, if “delta”, the method is “getGroupItemsPatch”, if “history”, the method is getHistories.
- depth=”integer”: if the preceding query is set to “history”, this query specifies the “depth” argument of the method getHistories.

The “Content-Type” response header must be a payload media types (i.e. not application/x.tacsit-link).

The “Last-Modified” request header must be set.

The payload of the response is either a GroupPayload or a list of EntityPayload or a list of EntityChangeSinkEvent or a list of EntityHistoryPayload depending on the method actually invoked (i.e. on the query fields), and using the PSM specified by the “Content-Type” response header.

The return code must be:

- 200 (OK): successful request;
- 204 (Empty): successful request, the result is empty;
- 304 (Not Modified): the “If-Modified-Since” request header was used and the resulting list did not change meanwhile; this is mandatory only for the getGroup and getGroupItems methods;

- 400 (Bad Request): a query field cannot be parsed;
- 404 (Not Found): UnknownURI or IsNotAGroup exceptions;
- 412 (Precondition Failed): the “If-Unmodified-Since” request header was used and the resulting list did change meanwhile; this is mandatory only for the getGroup and getGroupItems methods;
- 415 (Unsupported Media Type): the “Accept” request header is wrongly set;
- 5xx range: implementation-specific server problem.

11.3.3.4 URL: {baseURL}/entities/{id} – Verb: Get

In the URL, id is the identifier of a known entity.

This resource maps the following method:

- DataSink:getHistory(URI, integer).

The “Accept” request header must be set to any payload media types (i.e. not application/x.tacsit-link).

The “If-Modified-Since” and “If-Unmodified-Since” request headers may be set.

The request may have the following query field:

- entityQuery=”string”: the “filter” argument of the methods taking an EntityQuery argument; the mapping of such an EntityQuery to a string is specified in section 12.1.3.1;
- depth=”integer”: the “depth” argument of the method getHistory.

The “Content-Type” response header must be a payload media types (i.e. not application/x.tacsit-link).

The “Last-Modified” request header must be set.

The payload of the response is an HistoryEntityPayload using the PSM specified by the “Content-Type” response header.

The return code must be:

- 200 (OK): successful request;
- 204 (Empty): successful request, the result is empty;
- 304 (Not Modified): the “If-Modified-Since” request header was used and the resulting list did not change meanwhile; this code is not mandatory to implement;
- 400 (Bad Request): a query field cannot be parsed;
- 404 (Not Found): UnknownURI or IsNotAnEntity exceptions;
- 412 (Precondition Failed): the “If-Unmodified-Since” request header was used and the resulting list did change meanwhile; this code is not mandatory to implement;
- 415 (Unsupported Media Type): the “Accept” request header is wrongly set;
- 5xx range: implementation-specific server problem.

Annex A: Standardized Extension Schema (normative)

This annex introduces the standardised set ExtendedData by grouping them by ExtensionSchema as designed in the Payload PIM.

ExtensionSchema “AIS”

This schema holds the following ExtendedData:

- “callSign” (string): The radio call sign of the platform received on AIS;
- “IMONumber” (string): The international maritime organization number allows encyclopedic information to be looked up in database provided by organizations such as Lloyds;
- “MMSI” (string): The maritime mobile service identity, allocated to the ship with the radio license;
- “nationality”: The nationality of the platform as received on AIS;
- “beam” (Distance): The beam width of the ship - port to starboard;
- “length” (Distance): The length of a ship - bow to stern;
- “maximumDraught” (Distance): The maximum present static draught as reported on AIS in accordance with IMO Resolution A.851;
- “positionalFixMethod” (string): The method used to obtain the positional fix reported on AIS;
- “RAIMinUse” (boolean): Whether receiver autonomous integrity monitoring is in use;
- “rateOfTurn” (real): The rate of turn as reported by AIS.

Extension Schema “5516”

This schema holds the following ExtendedData:

- “hasSpecialProcessing” (boolean): The Special Processing indicator is set;
- “isAnEmergency” (boolean): The Emergency indicator is set;
- “isForceTell” (boolean): The ForceTell indicator is set;
- “isInExercise” (boolean): The Entity has an Exercise indicator and so is part of an exercise;
- “isOfSpecialInterest” (boolean): The Special Interest indicator is set;
- “isSimulation” (boolean): The Simulation indicator is set - e.g. the entity is for operator training purposes;
- “Strength” (integer): The estimated number of objects represented by the Entity;
- “trackState” (string): If a track-like entity the state or phase of its life cycle; Indicates whether tracking is active; possible values: DEAD_RECKONED, DELETED, LOST, TRACKED;
- “hasMode4” (boolean): The Mode IV Indicator is set;
- “mode1Code” (short): Mode I Code received;
- “mode2Code” (short): Mode II Code received;
- “mode3Code” (short): Mode III Code received;
- “className” (string): This is semantically equivalent to the platform specific type except for being unconstrained to being in the STANAG 5516 pre-defined list;
- “environment” (string): The environment that the Entity is in or to which it applies; possible values: AIR, LAND, SURFACE, SUBSURFACE, SPACE;
- “identity” (string): The standard identity of the (track-like) entity; possible values: PENDING, UNKNOWN, ASSUMED_FRIEND, NEUTRAL, SUSPECT, HOSTILE);

- “nationality”: The nationality or country of origin of the object represented by the Entity;
- “platformActivity” (short): The code for the per environment defined platform activity for the Entity according to STANAG 5516; The activity describes what the object that the Entity represents is doing;
- “platformIdentity” (short): The code for the per environment defined platform identify for the Entity according to STANAG 5516; If no identity is defined then the code refers to a value in the reference point tables within STANAG 5516;
- “platformSpecificType” (short): The per environment defined specific type for the Entity according to STANAG5516; This defines the manufacture or build type of the object represented by the Entity; E.g. B747, L-100 HERCULES, HALIFAX FF;
- “unitName” (string): The name of the actual object instance represented by the entity; E.g. ship name, flight id.

ExtensionSchema “ADS-B”

This schema holds the following ExtendedData:

- “barometricVerticalRate” (real): The rate of change of height / altitude of the broadcasting aircraft calculated by barometric means;
- “country” (string): The country (abbreviated) of the aircraft, broadcast on ADS-B;
- “flightID” (string): The flight ID of the aircraft, broadcast on ADS-B;
- “flightLevel” (Distance): The flight level of the broadcasting aircraft;
- “fullCountry (string): The country (unabbreviated) of the aircraft, broadcast on ADS-B;
- “isOnTheGround” (boolean): Whether the broadcasting platform is currently on the ground;
- “modeSAddress” (string): The mode S address of the aircraft, broadcast on ADS-B, as set for IFF responses.

ExtensionSchema “APP6B”

This is a placeholder for a future extension schema.

ExtensionSchema “APP6C”

This is a placeholder for a future extension schema.

ExtensionSchema “2525C”

This is a placeholder for a future extension schema.

ExtensionSchema “2525D”

This is a placeholder for a future extension schema.

Annex B: New PSM for the TCI Standard (normative)

DDS PSM

The DDS PSM defines a set of IDL files for the TCI PIM classes. The TCI PSM defines the following DDS topic types:

- EntityQuery,
- QueryResult,
- SelectionEvent,
- TACSITController,
- Viewport

The detailed rules for the MDA code generation from the Data Payload PIM to the DDS PSM IDL are as follows:

- The PIM attributes are mapped to IDL attributes;
- Optional attributes are mapped to a union type with a single member present when the exists case attribute is true;
- Collections in the PIM are mapped to IDL sequences;
- Specialization / Generalization PIM relationships are mapped to IDL unions (or enumerations in the case of attribute-less abstractions);
- PIM accessor operations (read/get) are mapped to IDL attributes;
- Listener interfaces are mapped to DDS DCPS;
- Manager classes are mapped to DDS DCPS.

C# PSM

The C# PSM maps the TCI PIM classes and interfaces to C# classes and interfaces using MDA code generation. The detailed rules for the code generation are as follows:

- PIM classes are mapped to C# classes;
- PIM interfaces are mapped to C# interfaces;
- Listener interfaces are mapped to the C# event mechanism; rate limited or unlimited events are accessed through a factory method;
- The PIM attributes are mapped to a C# class private member attributes (camel case) and a public read-write property (Pascal case);
- A public constructor is defined for all the mandatory attributes;
- Optional attributes that cannot take a null value (enumerations and scalar data-types) are mapped to an instantiation of the System.Nullable class;
- Ordered collections in the PIM are mapped to an instantiation of the IList interface;
- Unordered collections in the PIM are mapped to an instantiation of the IEnumerable interface;
- Specialization / Generalization PIM relationships are mapped to C# class inheritance.

TypeScript PSM

For now, this PSM only map the interfaces of TCI needed by TEX: the EntityType interface and the GeodeticPosition interface.

It is to be found in tacsit-tci-controller.d.ts that declares these two interfaces.