Date: August 2024

Tactical Decision Aids Interface (TDAI)

Version 1.0

OMG Document Number: formal/24-08-01 [smsc/24-08-01]

Standard document URL: https://www.omg.org/spec/TDAI/

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 https://www.omg.org, under Specifications, Report a Bug/Issue (https://issues.omg.org/issues/create-new-issue).

Table of Contents

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 https://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:

https://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
9C Medway Road, PMB 274
Milford, MA 01757
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 https://www.iso.org

Issues

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

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

1 Scope

This specification defines the interface between components of a C2 (Command and Control) system concerned with the dissemination of tactical picture information, recommendations for changes, refinements and enhancements to that picture and recommendations for courses of action that relate to the picture with resources for which the C2 system's user have responsibility. As such it is a specification for an interface between tactical picture management components and tactical decision aids, supporting the development of open modular C2 systems.

2 Conformance

This specification defines conformance points to promote both applicability and interoperability. The conformance points recognize the decomposition of the specification into services relating to Tactical Picture and services relating to Plan Execution. Services within the specification relating specifically to either Tactical Picture or Plan Execution are optional. The mandatory services within the interface are those in the AbstractRecommendations package. Conformance Points define a set of services to be implemented by a Tactical Picture or Plan Execution component and a dependency for a Tactical Decision Aid. Conformation points are defined for functional subsets, PSM technologies and PSM external standards.

Table 2.1 – Conformance Points for TDA

Conformance Point	Service Interfaces	Rationale
Functional		
Basic Tactical Picture	Configuration, Response, Categorization, PictureManagement	These interfaces include the types of tactical picture recommendation most likely to be made by decision aids.
Basic Plan Execution	Configuration, Response, PlanDataSink, ResourceDataSink, PlanExecutionAction, PlanExecutionControl	These interfaces include the types of plan execution recommendation most likely to be made by decision aids.
Extended Tactical Picture	As per Basic Tactical Picture plus RecommendationManagement, ExtendedCategorization, ExtendedPictureManagement	These are the additional interfaces for tactical picture recommendations also support for cancellation
Extended Plan Execution	As per Basic Plan Execution plus RecommendationManagement, ExtendedPlanExecutionAction, ExtendedPlanExecutionControl	These are the additional interfaces for plan execution recommendations also support for cancellation
PSM Technologies		
DDS	As defined by functional conformance points	A PSM technology for near real time operation
GraphQL	As defined by functional conformance points	A PSM for flexible data access
External Standards	**Schema Prefix**	
STANAG 5516	s5516.*	Naval applications
JC3IEDM	jc3iedm	Applications for joint operations

APP6	app6b, app6c	General C2 applications
SOPES	sopes	General C2 applications

3 Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

Table 3.1 – Normative References

Title (Acronym)	Version / Date	Organization	Reference / URL
TACSIT Data Exchange (TEX)	1.1 FTF / August 2023	OMG	dtc/23-08-02 www.omg.org/members/cgi-bin/doc?dtc/23-08-02
Data Distribution Service (DDS)	1.4 / March 2015	OMG	formal/2015-04-10 www.omg.org/spec/DDS
Interface Definition Language (IDL)	4.2 / January 2018	OMG	formal/2018-01-05 www.omg.org/spec/IDL
Extended View of Time (EVOT)	2.0 August 2008	OMG	formal/2008-08-01 www.omg.org/spec/EVOT
DDS Security	1.1 July 2018	OMG	formal/18-04-01 https://www.omg.org/spec/DDS-SECURITY/
Shared Operational Picture Exchange Services (SOPES)	1.0 May 2011	OMG	formal/11-05-04 www.omg.org/spec/SOPES
Graph Query Language (GraphQL)	October 2021	GraphQL Foundation	https://spec.graphql.org/October2021/
Quantities and units	November 2011	ISO	ISO 80000-1:2022 https://www.iso.org/standard/76921.html
NATO Tactical Data Exchange – Link 16	Edition 6	NATO	STANAG 5516
Joint C3 Information Exchange Data Model	Rev D CN 1	NATO	STANAG 5525
Joint C3 Information Exchange Data Model (JC3IEDM)	v3.1.4	NATO	
NATO Joint Military Symbology (APP-6(B))	June 2008	NATO	
NATO Joint Military Symbology (APP-6(C))	May 2011	NATO	
World Meteorological Organization (Sea State Code)	latest	WMO	https://community.wmo.int/en/activity-areas/imop/cimo-guide

4 Terms and Definitions

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

API	(Application Programming Interface)
APP	(Allied Procedural Publication)
C2	(Command and Control)
CMS	(Combat Management System)
CORBA	(Common Object Request Broker Architecture)
CWM	(Common Warehouse Metamodel)
DDS	(Data Distribution Service)
EVOT	(Enhanced View of Time)
GraphQL	(Graph Query Language)
IDL	(Interface Definition Language)
IIOP	(Internet Inter-Orb Protocol)
IPR	(Intellectual Property Right)
ICAO	(International Civil Aviation Organization)
ISO	(International Organization for Standardization)
IMO	(International Maritime Organization)
JC3IEDM	(Joint Consultation, Command and Control Information Exchange Data Model)
LOI	(Letter of Intent)
MDA	(Model Driven Architecture)
MOF	(Meta Object Facility)
NS	(Naming Service)
NATO	(North Atlantic Treaty Organization)
OARIS	(Open Architecture Radar Interface Standard)
ODF	(Open Document Format)
OMG	(Object Management Group)
PIM	(Platform Independent Model)
PSM	(Platform Specific Model)
SOA	(Service Oriented Architecture)
SoaML	(Service oriented architecture Modeling Language)
STANAG	(NATO Standardization Agreement)

TF	(Task Force)
UML	(Unified Modeling Language)
XMI	(XML Metadata Interchange)
XML	(eXtensible Markup Language)

5 Symbols

No special symbols are introduced in this specification.

6 Additional Information

6.1 Acknowledgments

The following companies submitted this specification:

- BAE Systems

7 Tactical Decision Aids Interface Overview

The goal of the Tactical Decision Aids Interface specification is to support the on-going need to extend and upgrade C2 Systems, particularly to meet growing demand for automation and the exploitation of machine intelligence. This specification addresses the need to insert functionality into command and control systems that supports the user's decision-making process.

To insert such functionality efficiently and affordably, C2 system integrators need the freedom to source such functionality from multiple potential providers in the form of modular applications. This is especially so for systems meeting complex requirements in the military domain. This specification enables such an approach by standardizing the interface for such functionality.

This specification defines a set of services and a supporting data model as C2 Systems typically have demanding assurance requirements and are qualified for use by an overall design authority based on evidence built up from the constituent components. The constituent components need to know, *a priori*, the standardized data-model and services the system uses, so that they are able to provide qualification evidence for overall system assurance.

Consistency of semantics across the definitions of services and the data model is important for the interoperation of different component implementations that meet the standard, enabling the modular solution sought. To this end, the specification provides meta-models for the services and data-model, which define super-classes for the data-model and common patterns for the information and recommendation services.

The specification is organized as follows:

- Section 8 contains the Meta-Model with packages for super-classes and other classifiers with broad applicability across the Data Models.
- Section 9 contains the Data-Models describing a Data Model for Plan Execution and an Extension of the TACSIT Data Exchange Data Model to support the Tactical Picture Data Model requirements.
- Section 10 the Service Models describing the Information and Recommendation Service Models for Tactical Picture and Pan Execution.
- Section 11 describes the Domain Model Platform Specific Models for DDS and GraphQL.
- Section 12 describes the Service Model Platform Specific Models for DDS and GraphQL.
- Section 13 describes the Platform Specific Models for Extensible Enumerations; this section standardizes the representation of categories defined in a set of external specifications, whilst providing an extension mechanism that supports additional specifications and system-specific concepts.

The specification is captured as an Enterprise Architect (EA) UML version 2.1 model; sections 8, 9 & 10 are automatically generated into the specification from the model.

7.1 Use of the Tactical Decision Aids Interface

This subsection provides non-normative outline usage of the interface specification for two simple use cases: creating a 'Classification App' that recommends classification categories of TACSIT Entities (e.g., Frigate, Helicopter, Truck or Submarine); creating a 'Plan Monitoring App' that recommends when to start and stop different elements of an overall plan. These simple use cases use the Tactical Picture and Plan Execution services separately, other use cases may use these services in combination. The Tactical Decision Aids Interface (TDAI) is designed to be used in conjunction with the TACSIT Data Exchange (TEX) specification; the principle linkage is the EntityRef datatype, which provides a reference to the entities published by the TEX DataSink interface.

7.1.1 Classification App use of the Tactical Decision Aids Interface

(Non-Normative)
An outline design of the Classification App to use the Tactical Decision Aids Interface is as follows:

- Connect to the TEX and TDAI interfaces of the Tactical Picture component using the appropriate PSM method and system-specific configuration.
- Use a PSM method to register implementation of the recommendationProcessed operation on the Response interface.
- Use the PSM mapping of the isSupported operation on the Configuration interface for recommendClassification to verify that the TacticalPicture implementation supports the relevant recommendation operation.
- Use the getSupportMapping operation on the Configuration interface to access the extensible enumeration mapping-file. Verify that there are classification categories in the mapping file corresponding to the categories in the App's business rules. E.g. if the App has a rule for distinguishing helicopters find a helicopter category in the mapping file. Note that it is an option to design the App with known categories in mind.
- Create and add an Entity Listener to the TEX DataSink interface.
- Use the TEX getSet operation on the DataSink interface to get an initial view of the TEX Entities in the Tactical Picture.
- Process the initial view of Entities and any changes (including new Entities) from the dataChanged callback on the listener with the App's business logic.
- For any Entities for which the App's business logic can offer an improved classification category, use the recommendClassification operation on the Categorization interface to recommend the category to the TacticalPicture component. Internal to the App, note that a recommendation for this Entity is in progress (do not send further recommendations whilst in-progress).
- Implement the recommendationProcessed operation to clear the in-progress indicator if accepted, clear and log explanation (error code) if rejected and optionally log if deferred.
- Continue until the App is stopped (system specific mechanism outside of the scope of this specification).

7.1.2 Plan Monitoring App use of the Tactical Decision Aids Interface

An outline design of the Plan Monitoring App to use the Tactical Decision Aids Interface is as follows:

- Connect to the TDAI interface of the Plan Execution component using the appropriate PSM method and system-specific configuration.
- Use a PSM method to register implementation of the recommendationProcessed operation on the Response interface.
- Use the PSM mapping of the isSupported operation on the Configuration interface for start and terminate to verify that the Plan Eecution implementation supports the relevant recommendation operation.
- Use the getSupportMapping operation on the Configuration interface to access the extensible enumeration mapping-file. Verify that there are plan constituent categories in the mapping file corresponding to the categories in the App's business rules. E.g. if the App has a rule for 'search and rescue' plans find a 'search and rescue' category in the mapping file. Note that it is an option to design the App with known categories in mind.
- Create and add a Plan Listener to the TDAI PlanDataSink interface.
- Use the TDAI getSet operation on the PlanDataSink interface to get an initial view of the TDAI PlanConstituents known to the system.
- Process the initial view of PlanConstituents and any changes (including new PlanConstituents) from the dataChanged callback on the listener with the App's business logic.

- For any PlanConstituents which the App's business logic suggests should be started, use the start operation on the PlanExecutionControl interface to recommend that the PlanConstiuent is started. Internal to the App, note that a recommendation for this PlanConstiuent is in progress (do not send further recommendations whilst in-progress).
- Similarly, for any PlanConstituents which the App's business logic suggests should be terminated, use the terminate operation on the PlanExecutionControl interface to recommend that the PlanConstiuent is terminated.
- Implement the recommendationProcessed operation to clear the in-progress indicator if accepted, clear and log explanation (error code) if rejected and optionally log if deferred.
- Continue until the App is stopped (system specific mechanism outside of the scope of this specification).

This page intentionally left blank.

8 MetaModel

Parent Package: tactical decision aids

Meta-Model containing super-classes, data-types, patterns and the generic forms of interfaces that have applicability across the Domain Model and Service Model requirements for the Tactical Decision Aids specification. This is an extension mechanism supporting the requirements of similar Domain Models and Service Models that could apply in the future or within a system specific context.

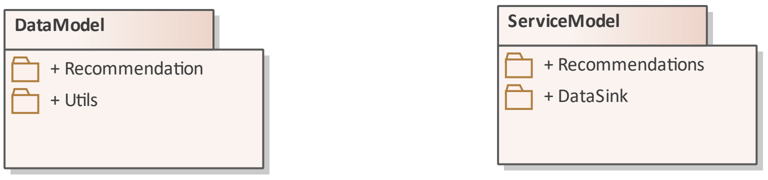


Figure 8.1 MetaModel (Package diagram)

8.1 DataModel

Parent Package: MetaModel

Aspects of the Meta-Model supporting the Domain Models

8.1.1 Recommendation

Parent Package: DataModel

The Recommendation package of the Data Model Meta-Model defines generic concepts to support system agnostic recommendations.

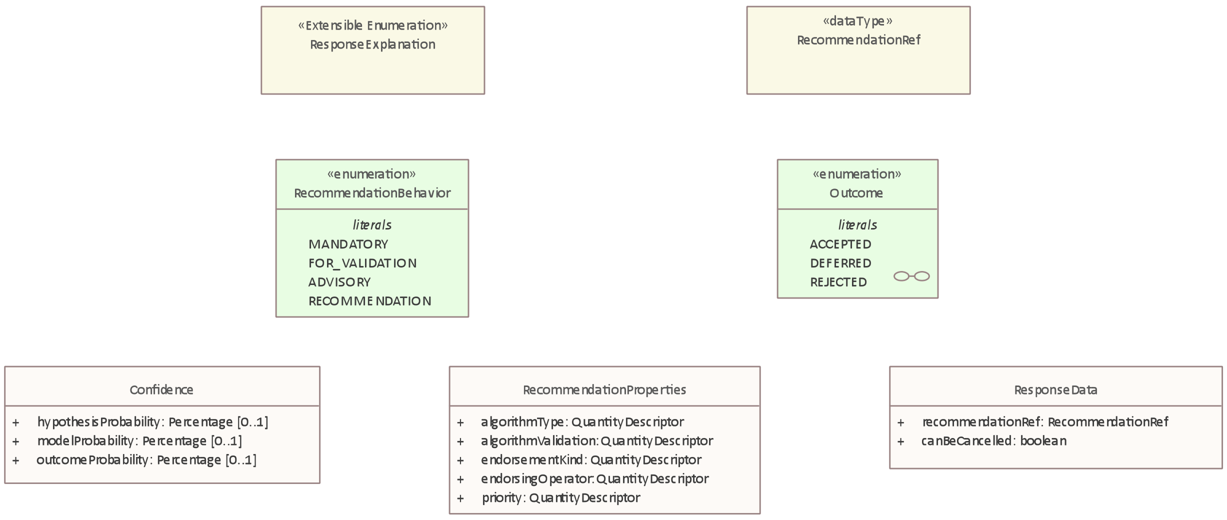

«Extensible Enumeration»
ResponseExplanation
«dataType»
RecommendationRef
«enumeration»
RecommendationBehavior
literals
MANDATORY
FOR_VALIDATION
ADVISORY
RECOMMENDATION
«enumeration»
Outcome
literals
ACCEPTED
DEFERRED
REJECTED
Confidence
+ hypothesisProbability: Percentage [0..1]
+ modelProbability: Percentage [0..1]
+ outcomeProbability: Percentage [0..1]
RecommendationProperties
+ algorithmType: QuantityDescriptor
+ algorithmValidation: QuantityDescriptor
+ endorsementKind: QuantityDescriptor
+ endorsingOperator: QuantityDescriptor
+ priority: QuantityDescriptor
ResponseData
+ recommendationRef: RecommendationRef
+ canBeCancelled: boolean

Figure 8.2 Recommendation and Response (Class diagram)

8.1.1.1 Confidence

Type: Class
Package: Recommendation
The statistical confidence placed in a recommendation

Table 8.1 – Attributes of Class Confidence

Attribute	Notes
hypothesisProbability Percentage [0..1]	The probability of the hypothesis associated with the Recommendation being true, given the model used as the Recommendation basis. This is the confidence that the Recommendation is correct or optimal according to the model.
modelProbability Percentage [0..1]	The probability of the model on which the Recommendation is based being applicable. This is the confidence in model given the particular model inputs.
outcomeProbability Percentage [0..1]	The probability that there will be a successful outcome from following a Recommendation according to the model used by the Recommendation. This attribute is only applicable to Recommendations that lead to actions on the external environment.

8.1.1.2 RecommendationProperties

Type: Class
Package: Recommendation
Additional extensible metadata relating to the recommendation process

Table 8.2 – Attributes of Class RecommendationProperties

Attribute	Notes
endorsingOperator QuantityDescriptor	An extensible categorization of a system user who has endorsed the recommendation.
endorsementKind QuantityDescriptor	An extensible categorization of the type endorsement made by the system user.
algorithmType QuantityDescriptor	An extensible categorization of type of algorithm used to make the recommendation.
algorithmValidation QuantityDescriptor	An extensible categorization of validation process that the algorithm has been subjected to.
priority QuantityDescriptor	An extensible categorization of the priority that a recipient should assign to a Recommendation.

8.1.1.3 ResponseData

Type: Class
Package: Recommendation
Additional information to describe the action actually performed. Specializations of this call allow the Decision Aid to find tactical picture updates and plan execution update corresponding to the recommendations. Binding this information to the response also means the Decision Aid doesn't need to store recommendation identifiers locally in order to perform post response processing.

Table 8.3 – Attributes of Class ResponseData

Attribute	Notes
recommendationRef RecommendationRef	A reference to the original recommendation that is unique across all clients in the system.
CanBeCancelled boolean	Indicates whether the service is able to process a subsequent cancelRecommendation request for this recommendation.

8.1.1.4 Outcome

Type: Enumeration
Package: Recommendation
The categories of outcome supported by the recommendation response interface.

Table 8.4 – Attributes of Enumeration Outcome

Attribute	Notes
«enum» **ACCEPTED**	The recommendation has been accepted and applied.
«enum» **DEFERRED**	The recommendation has been deferred, for instance for operator approval. An additional response will occur once a decision has been made.
«enum» **REJECTED**	The recommendation has been rejected. The explanation attribute contains any reason given for the rejection.

8.1.1.5 Recommendation Outcome

Type: StateMachine
Package: Recommendation

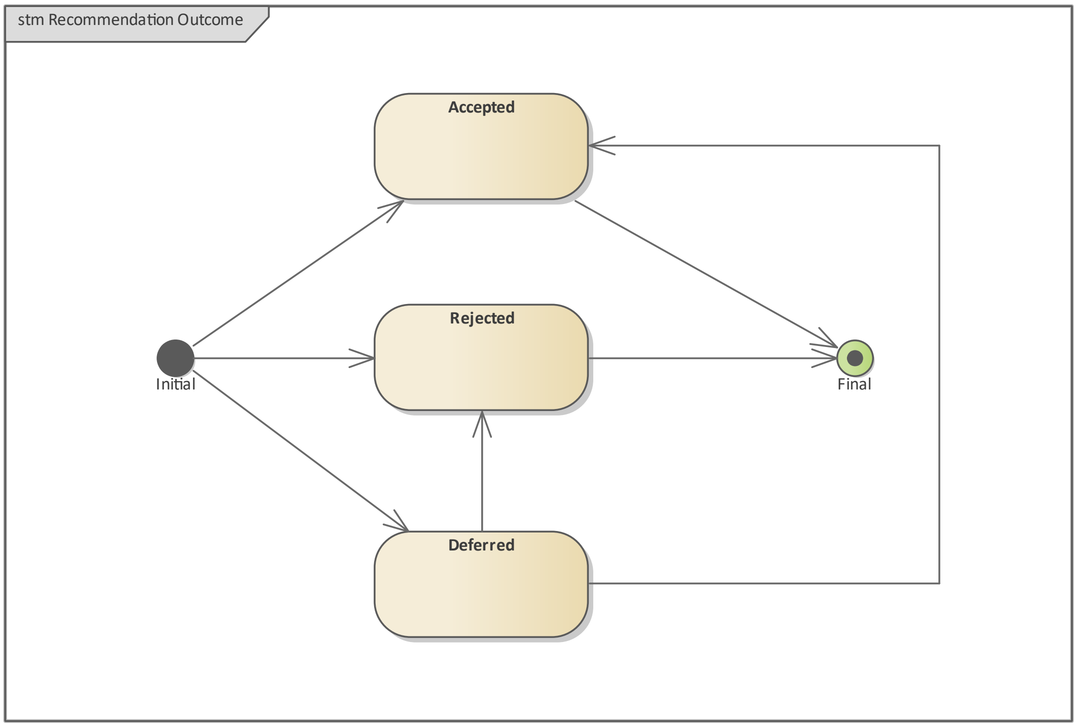


Figure 8.3 Recommendation Outcome (StateMachine diagram)
This diagram defines the state transitions of outcomes of recommendations from Tactical Decision Aids.

8.1.1.5.1 Accepted

Type: State
Package: Recommendation
The outcome of the recommendation is that is accepted and will be applied to the referenced instances.

8.1.1.5.2 Deferred

Type: State
Package: Recommendation
The initial outcome of the recommendation is that is deferred and will be either accepted or rejected after input from an operator or another system.

8.1.1.5.3 Rejected

Type: State
Package: Recommendation
The outcome of the recommendation is that is rejected and will not be applied to the referenced instances.

8.1.1.6 RecommendationBehavior

Type: Enumeration
Package: Recommendation
Categorization of Recommendations in terms of the recipient's behavior.

Table 8.5 – Attributes of Enumeration RecommendationBehavior

Attribute	Notes
«enum» **MANDATORY**	The Recommendation must be followed by the recipient.
«enum» **FOR_VALIDATION**	The recipient should enact subject to a confirmation process.
«enum» **ADVISORY**	The recommendation should be considered alongside alternative advisory sources of information.
«enum» **RECOMMENDATION**	No statement with respect to Recommendation categorization.

8.1.1.7 RecommendationRef

Type: DataType
Package: Recommendation
A reference to the recommendation that a Decision Aid has made. This must be unique within a system as a whole and not just within the lifetime of a decision aid or other system component.

8.1.1.8 ResponseExplanation

Type: DataType
Package: Recommendation
An explanation of the response to the recommendation. For example an error code.

8.1.2 Utils

Parent Package: DataModel
The Utils package in the Data Model Meta-Model defines utility classes required by other Data Model and Service Model packages.

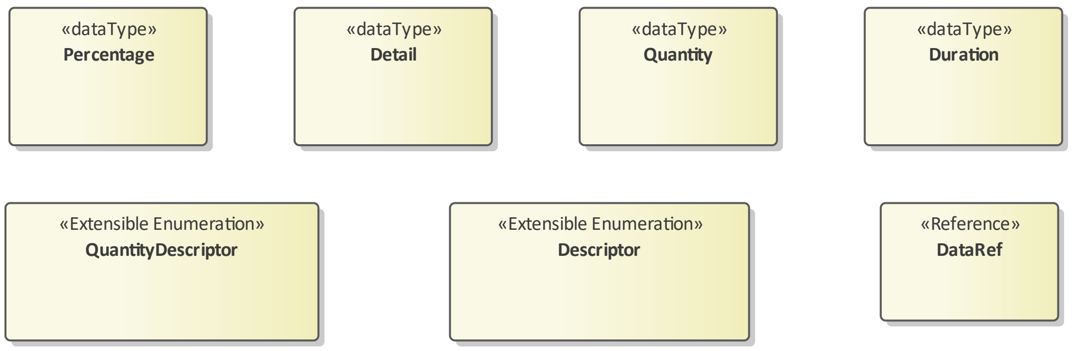


Figure 8.3 DataTypes (Class diagram)

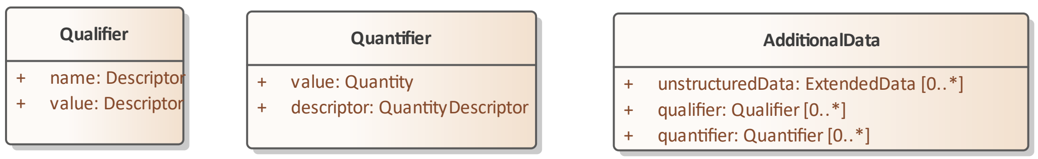

Qualifier
+ name: Descriptor
+ value: Descriptor
Quantifier
+ value: Quantity
+ descriptor: QuantityDescriptor
AdditionalData
+ unstructuredData: ExtendedData [0..*]
+ qualifier: Qualifier [0..*]
+ quantifier: Quantifier [0..*]

Figure 8.4 Structs (Class diagram)

8.1.2.1 AdditionalData

Type: Class
Package: Utils
Standardized encapsulation of qualitative, quantitative and unstructured data extension mechanisms.

Table 8.6 – Attributes of Class AdditionalData

Attribute	Notes
unstructuredData ExtendedData [0..*]	
qualifier Qualifier [0..*]	A set of additional qualitative attributes as an extension mechanism.
quantifier Quantifier [0..*]	A set of additional quantitative attributes as an extension mechanism.

8.1.2.2 Qualifier

Type: Class
Package: Utils
A class to represent additional, system-specific qualitative or categorical values as an extension mechanism.

Table 8.7 – Attributes of Class Qualifier

Attribute	Notes
name Descriptor	The name of quality being described
value Descriptor	The category value of the quality being described

8.1.2.3 Quantifier

Type: Class
Package: Utils

An abstract mechanism to quantify capabilities and dependencies

Table 8.8 – Attributes of Class Quantifier

Attribute	Notes
value Quantity	The numerical value of the concept being quantified
descriptor QuantityDescriptor	An extensible categorization of the type of quantity being specified. The descriptor is the determiner of the units (if any) associated with the quantity.

8.1.2.4 DataRef

Type: DataType
Package: Utils
A datatype with a platform specific mapping to represent a reference to a data item instance.

8.1.2.5 Descriptor

Type: DataType
Package: Utils
A general abstraction of categories to qualify an object. This class has an implementation specific null value that has the meaning of no statement being made in regard of the category of the descriptor.

8.1.2.6 Detail

Type: DataType
Package: Utils
This is a datatype with a platform specific mapping to represent additional information through an extension mechanism.

8.1.2.7 Duration

Type: DataType
Package: Utils
A datatype with a platform specific mapping to represent a relative length of time.

8.1.2.8 Percentage

Type: DataType
Package: Utils
A datatype with a platform specific mapping to represent a percentage value.

8.1.2.9 Quantity

Type: DataType
Package: Utils
A datatype with a platform specific mapping to represent a scalar quantity.

8.1.2.10 QuantityDescriptor

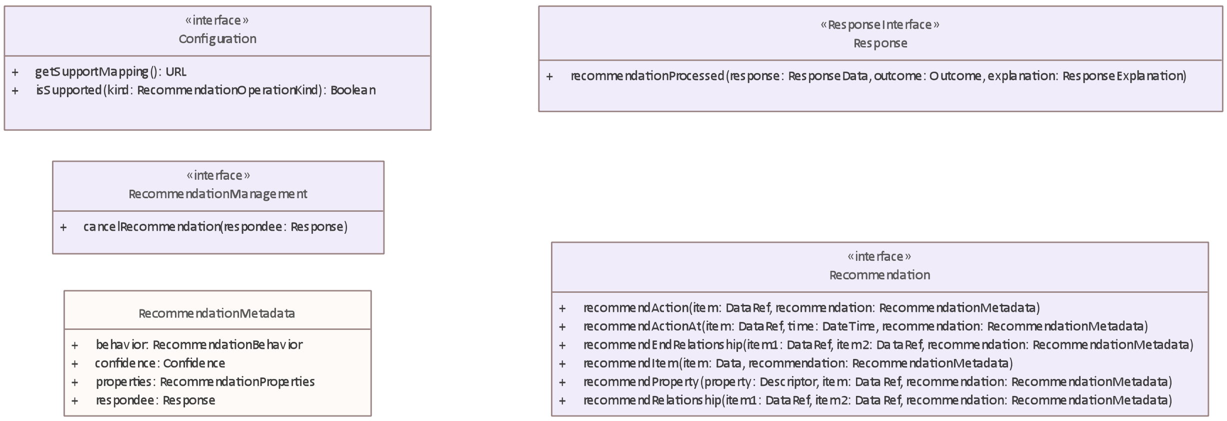
Type: DataType
Package: Utils
An abstraction of the categories of quantity.

8.2 ServiceModel

Parent Package: MetaModel
Aspects of the Meta-Model supporting the Service Models.

8.2.1 Recommendations

Parent Package: ServiceModel
This package defines the elements of the generic recommendation and response pattern used by Tactical Decision Aids to make recommendations.



«interface»
Configuration
+ getSupportMapping(): URL
+ isSupported(kind: RecommendationOperationKind): Boolean
«ResponseInterface»
Response
+ recommendationProcessed(response: ResponseData, outcome: Outcome, explanation: ResponseExplanation)
«interface»
RecommendationManagement
+ cancelRecommendation(respondee: Response)
RecommendationMetadata
+ behavior: RecommendationBehavior
+ confidence: Confidence
+ properties: RecommendationProperties
+ respondee: Response
«interface»
Recommendation
+ recommendAction(item: DataRef, recommendation: RecommendationMetadata)
+ recommendActionAt(item: DataRef, time: DateTime, recommendation: RecommendationMetadata)
+ recommendEndRelationship(item1: DataRef, item2: DataRef, recommendation: RecommendationMetadata)
+ recommendItem(item: Data, recommendation: RecommendationMetadata)
+ recommendProperty(property: Descriptor, item: DataRef, recommendation: RecommendationMetadata)
+ recommendRelationship(item1: DataRef, item2: DataRef, recommendation: RecommendationMetadata)

Figure 8.5 Recommendations (Class diagram)

8.2.1.1 RecommendationMetadata

Type: Class
Package: Recommendations
Additional information to describe and qualify all recommendations

Table 8.9 – Attributes of Class RecommendationMetadata

Attribute	Notes
confidence Confidence	The statistical confidence in the Recommendation
behavior RecommendationBehavior	The behavior required of the recipient
properties RecommendationProperties	Additional extensible properties of the Recommendation
respondee Response	The interface instance to which responses to the recommendation should be directed

8.2.1.2 Configuration

Type: Interface
Package: Recommendations
This interface allows clients (tactical decision aids) to determine system support for the interface in terms of extensibility methods and the operations implemented.

Table 8.10 – Methods of Interface Configuration

Method	Notes	Parameters
getSupportMapping()	This operation returns the location of a resource defining the implementing component's support for extensible enumeration values. The resource defines supported extensible enumeration values, maps them to extensible enumeration datatypes defined by this specification and external specifications from which they are derived.	
isSupported()	This operation defines whether a particular recommendation function is implemented in the system.	RecommendationOperationKind **kind** The operation for which support is being queried

8.2.1.3 Recommendation

Type: Interface
Package: Recommendations
The generic form of a recommendation interface with prototype recommendation operations.

Table 8.11 – Methods of Interface Recommendation

Method	Notes	Parameters
recommendProperty()	The prototype form of an operation to recommend a value for the property of an item.	Descriptor **property** DataRef **item** RecommendationMetadata **recommendation**
recommendAction()	The prototype form of an operation to recommend performing an operation on an item.	DataRef **item** RecommendationMetadata **recommendation**
recommendActionAt()	The prototype form of an operation to recommend performing an	DataRef **item** DateTime **time**

	operation on an item at a future time.	RecommendationMetadata **recommendation**
recommendRelationship()	The prototype form of an operation to recommend the creation of a relationship between two data items.	DataRef **item1** DataRef **item2** RecommendationMetadata **recommendation**
recommendEndRelationship()	The prototype form of an operation to recommend the ending of a relationship between two data items.	DataRef **item1** DataRef **item2** RecommendationMetadata **recommendation**
recommendItem()	The prototype form of an operation to recommend the creation of an item.	Data **item** RecommendationMetadata **recommendation**

8.2.1.4 Response

Type: Interface
Package: Recommendations
This interface is implemented by a tactical decision aid in order to receive responses to its recommendations. Each response operation contains a reference to the information contained in the corresponding recommendation.

Table 8.12 – Methods of Interface Response

Method	Notes	Parameters
recommendationProcessed()	This callback operation is invoked on the Tactical Decision Aid when the recommendation that it has made is accepted, deferred or rejected. This allows the Tactical Decision Aid to understand when a recommendation is in progress and to avoid redundantly repeating recommendations. There is one invocation of this callback per recommendation unless the first Outcome is Deferred, in which case there are two. This is shown in the Recommendation Outcome State Machine diagram.	ResponseData **response** Additional contextual and qualification data for the response. Outcome **outcome** The outcome of the recommendation process ResponseExplanation **explanation** Where available, an explanation of the recommendation processing, such as a reason for rejection.

8.2.2 DataSink

Parent Package: ServiceModel
The package defines the pattern for Tactical Decision Aids to receive Information on the instances of a particular class of data item.

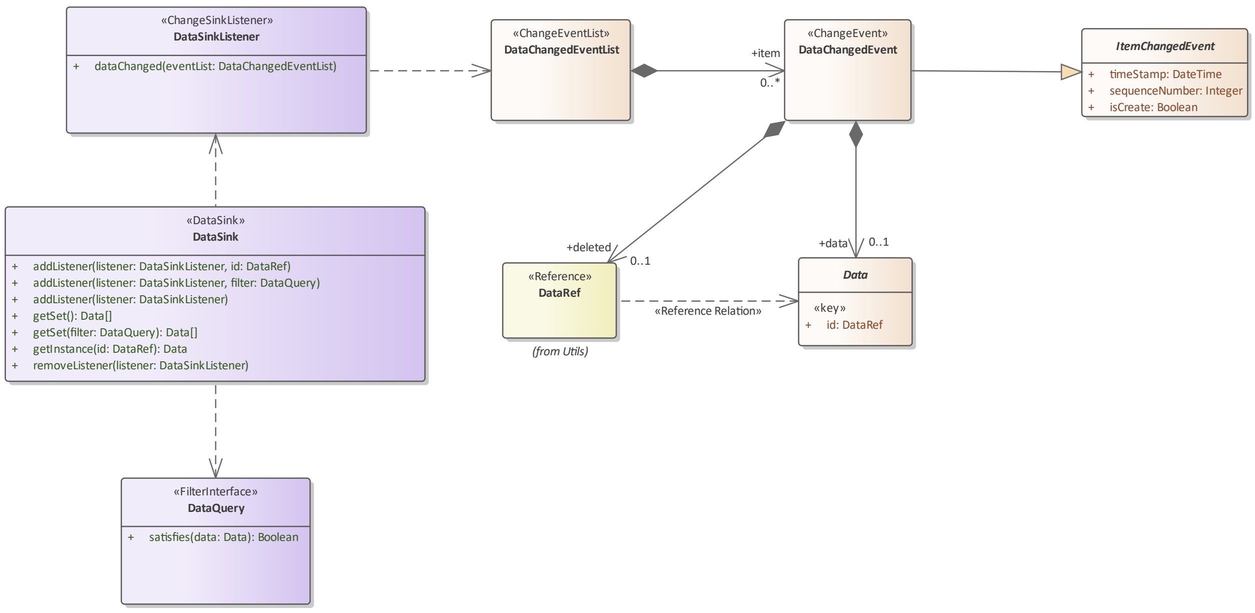

«ChangeSinkListener» DataSinkListener
+ dataChanged(eventList: DataChangedEventList)
«ChangeEventList» DataChangedEventList
«ChangeEvent» DataChangedEvent
+item
0..*
ItemChangedEvent
+ timeStamp: DateTime
+ sequenceNumber: Integer
+ isCreate: Boolean
«DataSink» DataSink
+ addListener(listener: DataSinkListener, id: DataRef)
+ addListener(listener: DataSinkListener, filter: DataQuery)
+ addListener(listener: DataSinkListener)
+ getSet(): Data[]
+ getSet(filter: DataQuery): Data[]
+ getInstance(id: DataRef): Data
+ removeListener(listener: DataSinkListener)
+deleted
0..1
+data
0..1
«Reference» DataRef
(from Utils)
«Reference Relation»
Data
«key»
+ id: DataRef
«FilterInterface» DataQuery
+ satisfies(data: Data): Boolean

Figure 8.6 DataSinkPattern (Class diagram)

8.2.2.1 Data

Type: Class
Package: DataSink
Represents the primary class of data for the Data Sink. Data items are the Data Sink's atomic unit.

Table 8.13 – Attributes of Class Data

Attribute	Notes
«key» **id** DataRef	Unique reference for the data item within the scope of the system.

8.2.2.2 DataChangedEvent

Type: Class
Package: DataSink
Represents information about a change to a <Data Instance>

8.2.2.3 DataChangedEventList

Type: Class
Package: DataSink
Represents the list of changes to <Data> since the last event notified to that instance of the listener. Multiple changes may be consolidated into a single callback to a listener on the interface.

8.2.2.4 ItemChangedEvent

Type: Class
Package: DataSink
An abstraction of data sink change event

Table 8.14 – Attributes of Class ItemChangedEvent

Attribute	Notes
timeStamp DateTime	The time of the change
sequenceNumber Integer	The time sequenced position of the event for the listener
isCreate Boolean	The event created a new instance

8.2.2.5 DataQuery

Type: Interface
Package: DataSink
This is an interface through which a client can define Queries on <Data> so as to filter the information returned. Classes implementing the interface provide means to set the query parameters (such as a constructor).

Table 8.15 – Methods of Interface DataQuery

Method	Notes	Parameters
satisfies()	This operation is the client's implementation of a filtering query for <Data>	Data **data** The data being filtered

8.2.2.6 DataSink

Type: Interface
Package: DataSink
This interface contains operations that give a Tactical Decision Aid access to information about the execution of <Data>. A Tactical Decision Aid can add and remove listeners as well as reading the information about individual <Data> or all or a filtered subset of <Data>.

Table 8.16 – Methods of Interface DataSink

Method	Notes	Parameters
addListener()	Operation to add a listener for callbacks relating to a single <Data Instance>	DataSinkListener **listener** The listener object to receive the callback DataRef **id** A reference to a specific instance of interest
addListener()	Operation to add a listener for callbacks relating to all <Data> that satisfy the Query	DataSinkListener **listener** The listener object to receive the callback DataQuery **filter** The filer object to apply to changes
addListener()	Operation to add a listener for callbacks relating to all <Data>	DataSinkListener **listener** The listener object to receive the callback
getSet()	Operation to obtain the information relating to all the <Data>	
getSet()	Operation to obtain the information relating to all the <Data> satisfying the query	DataQuery **filter** The filter object to apply to instance of the class
getInstance()	Operation to obtain the information relating to the <Data> reference	DataRef **id** A reference to the speciifc instance of interest
removeListener()	Operation to remove a listener	DataSinkListener **listener** The listener object to no longer receive callbacks

8.2.2.7 DataSinkListener

Type: Interface
Package: DataSink
This is an interface for clients to implement callback to receive information on changes to <Data>.

Table 8.17 – Methods of Interface DataSinkListener

Method	Notes	Parameters
dataChanged()	This operation is implemented by the client to process the data change callback. Multiple changes can be notified through a single invocation.	DataChangedEventList **eventList** The list of data instances that have changed

9 DataModel

Parent Package: tactical decision aids
The Tactical Decision Aids Data Model defines the representation of information that is passed between Picture Management Components and Tactical Decision Aid Components.

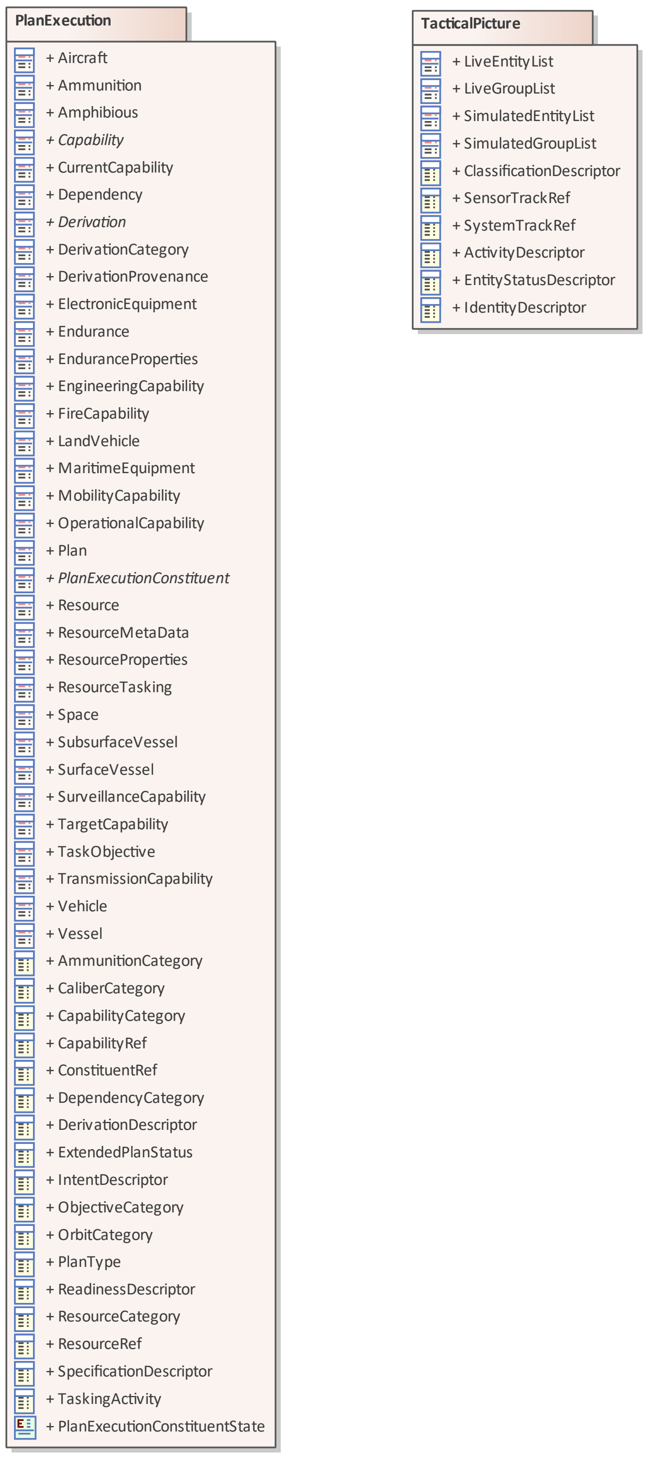
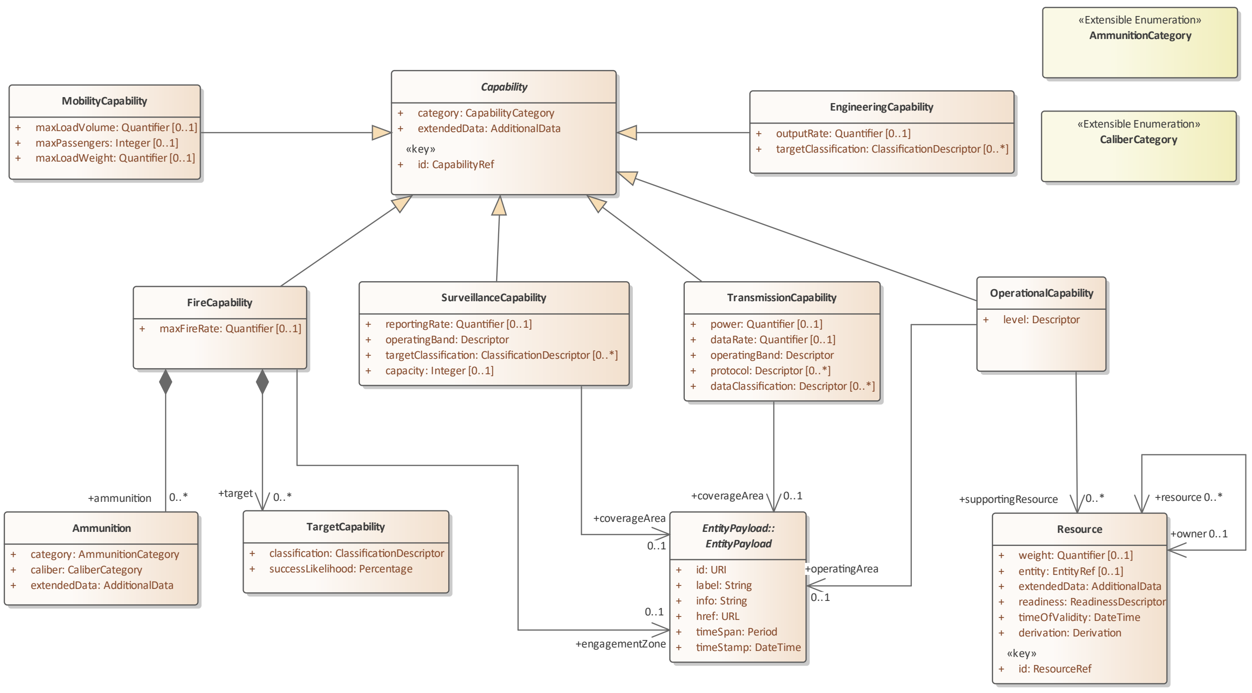


Figure 9.1 DataModel (Package diagram)

9.1 PlanExecution

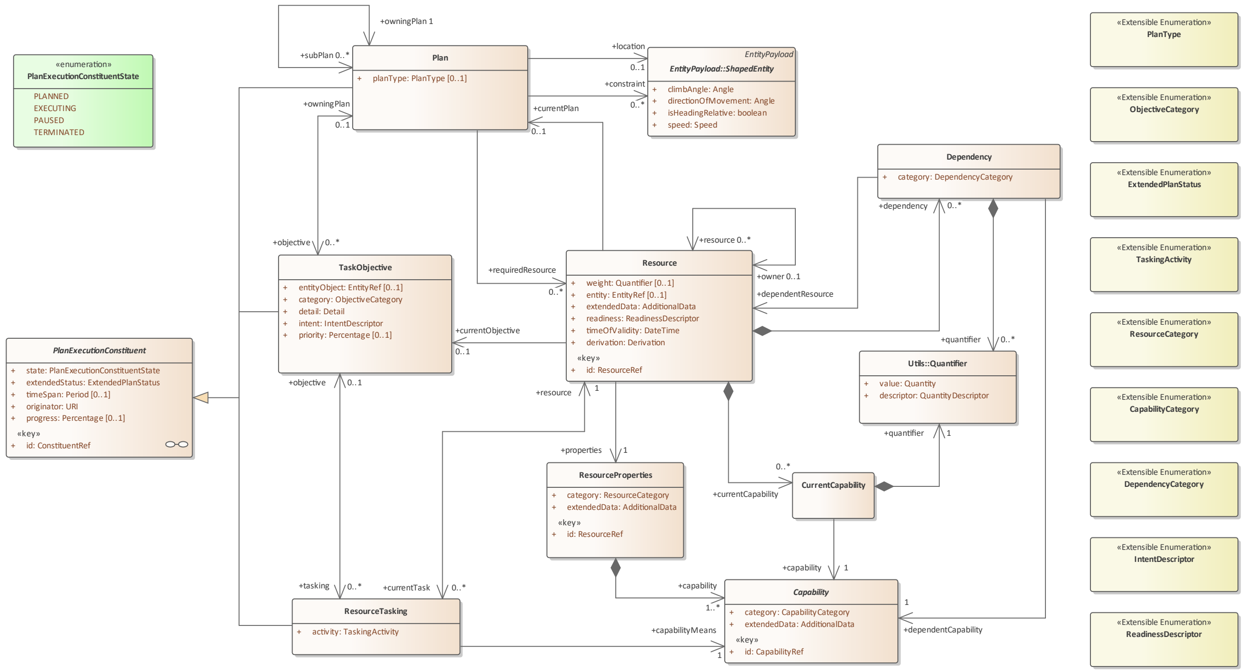
Parent Package: DataModel

The Plan Execution package of the Data Model defines the generic concepts necessary to recommend, execute and amend tactical plans. Domain and system specific concepts are abstracted using the Descriptor extension mechanism.



«Extensible Enumeration» AmmunitionCategory
«Extensible Enumeration» CaliberCategory
MobilityCapability
+ maxLoadVolume: Quantifier [0..1]
+ maxPassengers: Integer [0..1]
+ maxLoadWeight: Quantifier [0..1]
Capability
+ category: CapabilityCategory
+ extendedData: AdditionalData
«key»
+ id: CapabilityRef
EngineeringCapability
+ outputRate: Quantifier [0..1]
+ targetClassification: ClassificationDescriptor [0..*]
FireCapability
+ maxFireRate: Quantifier [0..1]
SurveillanceCapability
+ reportingRate: Quantifier [0..1]
+ operatingBand: Descriptor
+ targetClassification: ClassificationDescriptor [0..*]
+ capacity: Integer [0..1]
TransmissionCapability
+ power: Quantifier [0..1]
+ dataRate: Quantifier [0..1]
+ operatingBand: Descriptor
+ protocol: Descriptor [0..*]
+ dataClassification: Descriptor [0..*]
OperationalCapability
+ level: Descriptor
+ammunition 0..*
+target 0..*
+coverageArea 0..1
+coverageArea 0..1
+supportingResource 0..*
+resource 0..*
+owner 0..1
Ammunition
+ category: AmmunitionCategory
+ caliber: CaliberCategory
+ extendedData: AdditionalData
TargetCapability
+ classification: ClassificationDescriptor
+ successLikelihood: Percentage
EntityPayload:: EntityPayload
+ id: URI
+ label: String
+ info: String
+ href: URL
+ timeSpan: Period
+ timeStamp: DateTime
+operatingArea 0..1
0..1 +engagementZone
Resource
+ weight: Quantifier [0..1]
+ entity: EntityRef [0..1]
+ extendedData: AdditionalData
+ readiness: ReadinessDescriptor
+ timeOfValidity: DateTime
+ derivation: Derivation
«key»
+ id: ResourceRef

Figure 9.2 Capability (Class diagram)



«enumeration» PlanExecutionConstituentState
PLANNED
EXECUTING
PAUSED
TERMINATED
+owningPlan 1
+subPlan 0..*
Plan
+ planType: PlanType [0..1]
+location 0..1
+constraint 0..*
EntityPayload
EntityPayload::ShapedEntity
+ climbAngle: Angle
+ directionOfMovement: Angle
+ isHeadingRelative: boolean
+ speed: Speed
+owningPlan 0..1
+currentPlan 0..1
Dependency
+ category: DependencyCategory
+dependency 0..*
+objective 0..*
TaskObjective
+ entityObject: EntityRef [0..1]
+ category: ObjectiveCategory
+ detail: Detail
+ intent: IntentDescriptor
+ priority: Percentage [0..1]
+requiredResource 0..*
+resource 0..*
Resource
+ weight: Quantifier [0..1]
+ entity: EntityRef [0..1]
+ extendedData: AdditionalData
+ readiness: ReadinessDescriptor
+ timeOfValidity: DateTime
+ derivation: Derivation
«key»
+ id: ResourceRef
+owner 0..1
+dependentResource
+currentObjective 0..1
+quantifier 0..*
PlanExecutionConstituent
+ state: PlanExecutionConstituentState
+ extendedStatus: ExtendedPlanStatus
+ timeSpan: Period [0..1]
+ originator: URI
+ progress: Percentage [0..1]
«key»
+ id: ConstituentRef
+objective 0..1
+resource 1
Utils::Quantifier
+ value: Quantity
+ descriptor: QuantityDescriptor
+quantifier 1
+properties 1
ResourceProperties
+ category: ResourceCategory
+ extendedData: AdditionalData
«key»
+ id: ResourceRef
0..* +currentCapability
CurrentCapability
+capability 1
+tasking 0..*
+currentTask 0..*
+capability 1..*
Capability
+ category: CapabilityCategory
+ extendedData: AdditionalData
«key»
+ id: CapabilityRef
1 +dependentCapability
ResourceTasking
+ activity: TaskingActivity
+capabilityMeans 1
«Extensible Enumeration» PlanType
«Extensible Enumeration» ObjectiveCategory
«Extensible Enumeration» ExtendedPlanStatus
«Extensible Enumeration» TaskingActivity
«Extensible Enumeration» ResourceCategory
«Extensible Enumeration» CapabilityCategory
«Extensible Enumeration» DependencyCategory
«Extensible Enumeration» IntentDescriptor
«Extensible Enumeration» ReadinessDescriptor

Figure 9.3 PlanExecution (Class diagram)

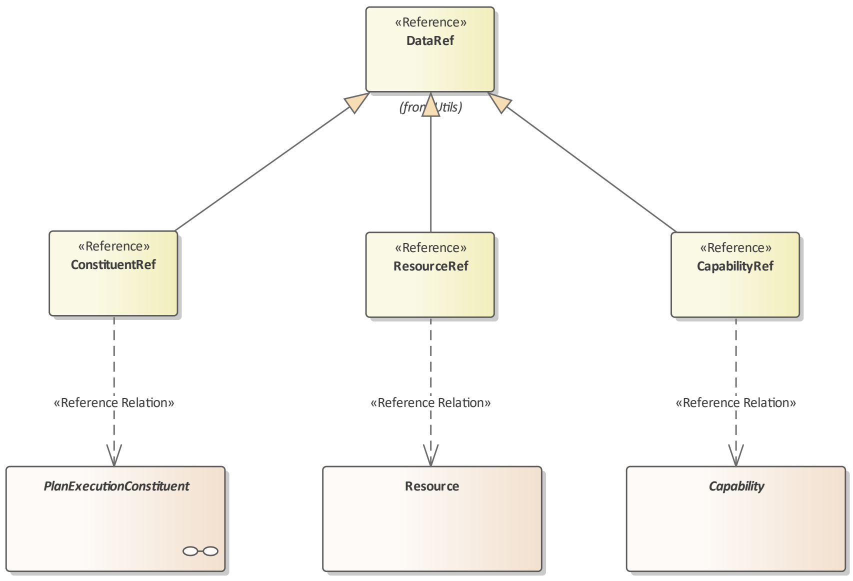


Figure 9.4 Reference DataTypes (Class diagram)

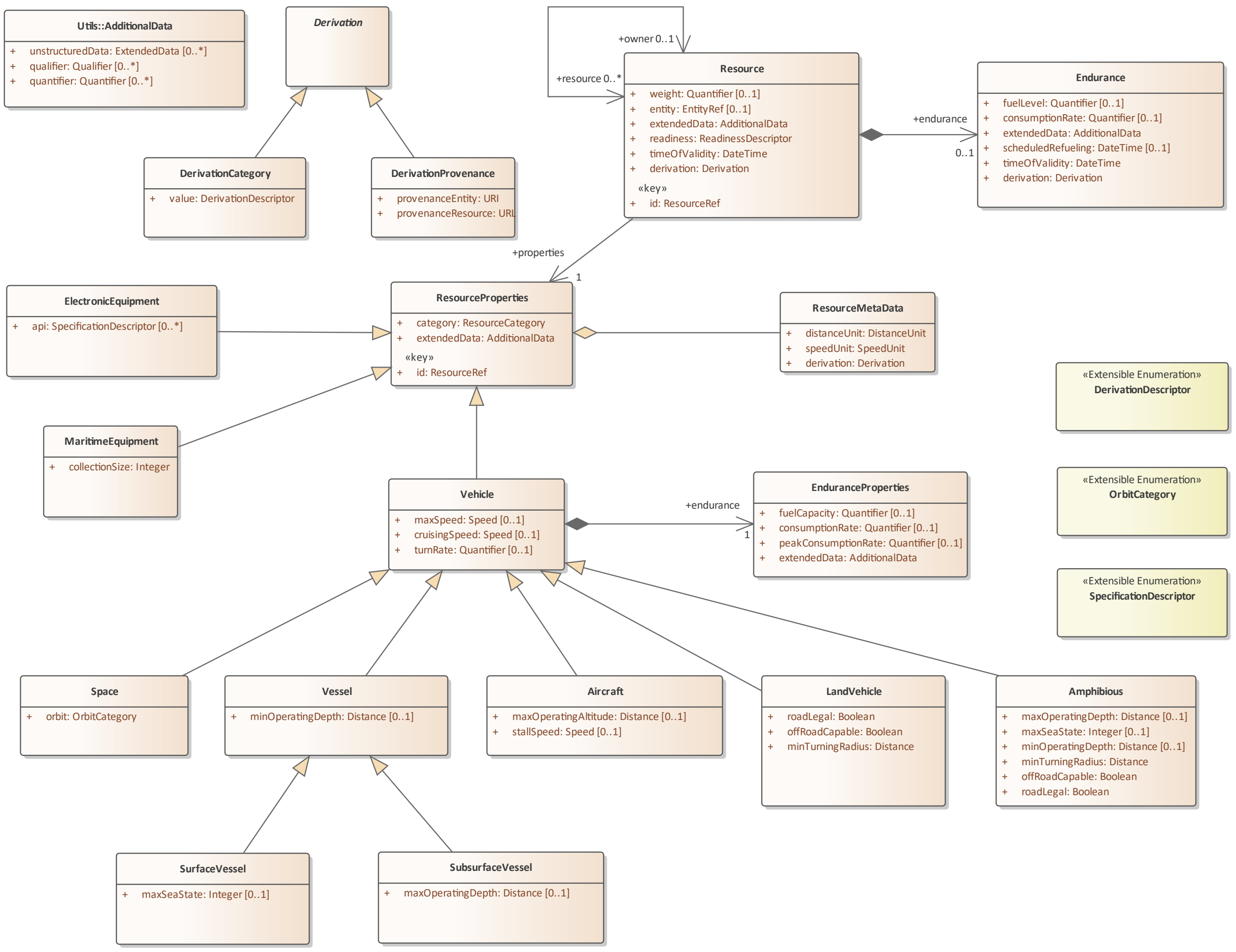

Utils::AdditionalData
+ unstructuredData: ExtendedData [0..*]
+ qualifier: Qualifier [0..*]
+ quantifier: Quantifier [0..*]
Derivation
DerivationCategory
+ value: DerivationDescriptor
DerivationProvenance
+ provenanceEntity: URI
+ provenanceResource: URL
+owner 0..1
+resource 0..*
Resource
+ weight: Quantifier [0..1]
+ entity: EntityRef [0..1]
+ extendedData: AdditionalData
+ readiness: ReadinessDescriptor
+ timeOfValidity: DateTime
+ derivation: Derivation
«key»
+ id: ResourceRef
+endurance
0..1
Endurance
+ fuelLevel: Quantifier [0..1]
+ consumptionRate: Quantifier [0..1]
+ extendedData: AdditionalData
+ scheduledRefueling: DateTime [0..1]
+ timeOfValidity: DateTime
+ derivation: Derivation
+properties
1
ElectronicEquipment
+ api: SpecificationDescriptor [0..*]
ResourceProperties
+ category: ResourceCategory
+ extendedData: AdditionalData
«key»
+ id: ResourceRef
ResourceMetaData
+ distanceUnit: DistanceUnit
+ speedUnit: SpeedUnit
+ derivation: Derivation
«Extensible Enumeration»
DerivationDescriptor
MaritimeEquipment
+ collectionSize: Integer
«Extensible Enumeration»
OrbitCategory
Vehicle
+ maxSpeed: Speed [0..1]
+ cruisingSpeed: Speed [0..1]
+ turnRate: Quantifier [0..1]
+endurance
1
EnduranceProperties
+ fuelCapacity: Quantifier [0..1]
+ consumptionRate: Quantifier [0..1]
+ peakConsumptionRate: Quantifier [0..1]
+ extendedData: AdditionalData
«Extensible Enumeration»
SpecificationDescriptor
Space
+ orbit: OrbitCategory
Vessel
+ minOperatingDepth: Distance [0..1]
Aircraft
+ maxOperatingAltitude: Distance [0..1]
+ stallSpeed: Speed [0..1]
LandVehicle
+ roadLegal: Boolean
+ offRoadCapable: Boolean
+ minTurningRadius: Distance
Amphibious
+ maxOperatingDepth: Distance [0..1]
+ maxSeaState: Integer [0..1]
+ minOperatingDepth: Distance [0..1]
+ minTurningRadius: Distance
+ offRoadCapable: Boolean
+ roadLegal: Boolean
SurfaceVessel
+ maxSeaState: Integer [0..1]
SubsurfaceVessel
+ maxOperatingDepth: Distance [0..1]

Figure 9.5 Resource (Class diagram)

9.1.1 Aircraft

Type: Class
Package: PlanExecution
An airborne resource

Table 9.1 – Attributes of Class Aircraft

Attribute	Notes
maxOperatingAltitude Distance [0..1]	The maximum altitude (barometric) at which the aircraft is able to operate.
stallSpeed Speed [0..1]	The minimum speed for the aircraft resource to operate below which it risks a stall.

9.1.2 Ammunition

Type: Class
Package: PlanExecution
Description of the ammunition associated with a fire capability. Here, ammunition generalizes to include bombs, torpedoes, decoys and missiles.

Table 9.2 – Attributes of Class Ammunition

Attribute	Notes
category AmmunitionCategory	An extensible categorization of the ammunition type
caliber CaliberCategory	An extensible description of the ammunition caliber (i.e. size).
extendedData AdditionalData	An extensibility mechanism for system specific ammunition attributes.

9.1.3 Amphibious

Type: Class
Package: PlanExecution

Table 9.3 – Attributes of Class Amphibious

Attribute	Notes
maxOperatingDepth Distance [0..1]	The operating limit of the amphibious vehicle with respect to depth in the water
maxSeaState Integer [0..1]	The World Meteorological Organization (WMO) sea state code - range 0 .. 9.
minOperatingDepth Distance [0..1]	The amphibious resource's operating limit with respect to shallow water
minTurningRadius Distance	The radius of the amphibious vehicle's tightest turning circle.
offRoadCapable Boolean	Whether the amphibious vehicle can be driven off-road. More granular off-road capabilities are specified using the mobility capability specialization.
roadLegal Boolean	The amphibious vehicle is legal for driving on the roads (in the territory applicable to the current system context).

9.1.4 Capability

Type: Class
Package: PlanExecution
A Capability is an abstraction of a Resource's fundamental properties with respect to its ability to undertake tasks.

Table 9.4 – Attributes of Class Capability

Attribute	Notes
category CapabilityCategory	An extensible categorization of the kind of capability being described.
extendedData AdditionalData	An extensibility mechanism for system specific capability attributes.
«key» **id** CapabilityRef	The unique identifier for the instance.

9.1.5 CurrentCapability

Type: Class
Package: PlanExecution

9.1.6 Dependency

Type: Class
Package: PlanExecution
This class represents a dependent linkage between two resources or of a resource on a particular capability.

Table 9.5 – Attributes of Class Dependency

Attribute	Notes
category DependencyCategory	This is an extensible categorization of the type of dependency.

9.1.7 Derivation

Type: Class
Package: PlanExecution
An abstract class for derivation metadata

9.1.8 DerivationCategory

Type: Class
Package: PlanExecution
A system specific categorical derivation

Table 9.6 – Attributes of Class DerivationCategory

Attribute	Notes
value DerivationDescriptor	

9.1.9 DerivationProvenance

Type: Class
Package: PlanExecution
Derivation conforming to the W3C PROV recommendation

Table 9.7 – Attributes of Class DerivationProvenance

Attribute	Notes
provenanceEntity URI	The entity pertaining to the derivation in the corresponding provenance resource, conforming to the W3C PROV recommendation.
provenanceResource URL	The provenance resource conforming to the W3C PROV recommendation, containing the provenance metadata for the derivation.

9.1.10 ElectronicEquipment

Type: Class
Package: PlanExecution
A resource whose primary capabilities relate to its electronic components.

Table 9.8 – Attributes of Class ElectronicEquipment

Attribute	Notes
api SpecificationDescriptor [0..*]	The set of interfaces standards that the equipment supports through which integrated functionality can be delivered to Plan Execution.

9.1.11 Endurance

Type: Class
Package: PlanExecution
This class encapsulates the dynamic endurance properties of a resource.

Table 9.9 – Attributes of Class Endurance

Attribute	Notes
fuelLevel Quantifier [0..1]	The current quantity of fuel available.
consumptionRate Quantifier [0..1]	The current rate at which the fuel is consumed.
extendedData AdditionalData	Additional dynamic information related to the Resource's Endurance.
scheduledRefueling DateTime [0..1]	The time at which more fuel is scheduled to be available.
timeOfValidity DateTime	The time for which the attributes of the Endurance class are valid.
derivation Derivation	A description of the means by which the data for the Resource's Endurance attributes were derived. This includes sensing, communication routes and human input.

9.1.12 EnduranceProperties

Type: Class
Package: PlanExecution
This class encapsulates the static, persistent endurance properties of a resource.

Table 9.10 – Attributes of Class EnduranceProperties

Attribute	Notes
fuelCapacity Quantifier [0..1]	The maximum quantity of fuel that can be stored.
consumptionRate Quantifier [0..1]	The nominal or mean (as defined by the descriptor) rate at which the fuel is consumed.
peakConsumptionRate Quantifier [0..1]	The peak rate of fuel consumption.
extendedData AdditionalData	Additional static information related to the Resource's Endurance.

9.1.13 EngineeringCapability

Type: Class
Package: PlanExecution
A capability to build, maintain, breach or demolish structures or infrastructure in the operational environment.

Table 9.11 – Attributes of Class EngineeringCapability

Attribute	Notes
outputRate Quantifier [0..1]	The nominal rate at which the engineering capability can be delivered.
targetClassification ClassificationDescriptor [0..*]	The categories of object to which the Engineering Capability can be applied.

9.1.14 FireCapability

Type: Class
Package: PlanExecution
An ability to apply physical effect towards an adversary.

Table 9.12 – Attributes of Class FireCapability

Attribute	Notes
maxFireRate Quantifier [0..1]	

9.1.15 LandVehicle

Type: Class
Package: PlanExecution
A vehicle primarily for traveling on land

Table 9.13 – Attributes of Class LandVehicle

Attribute	Notes
roadLegal Boolean	The land vehicle is legal for driving on the roads (in the territory applicable to the current system context).
offRoadCapable Boolean	Whether the land vehicle can be driven off-road. More granular off-road capabilities are specified using the mobility capability specialization.
minTurningRadius Distance	The radius of the land vehicle's tightest turning circle.

9.1.16 MaritimeEquipment

Type: Class
Package: PlanExecution
Equipment to be deployed in the maritime environment from surface or subsurface vessels. Maritime equipment do not encompass an entire vessel. The inherited owner-resource self-association relation from the parent Resource class applies between the Vessel and MaritimeEquipment classes.

Table 9.14 – Attributes of Class MaritimeEquipment

Attribute	Notes
collectionSize Integer	Where a resource denotes a set of items this attribute specifies how many there are. A default of 1 is used for single items.

9.1.17 MobilityCapability

Type: Class
Package: PlanExecution
An ability to transport objects and personnel.

Table 9.15 – Attributes of Class MobilityCapability

Attribute	Notes
maxLoadVolume Quantifier [0..1]	The maximum load by volume (including passengers) that the Mobility Capability can transport.

Attribute	Notes
maxPassengers Integer [0..1]	The maximum number of passengers that can be transported.
maxLoadWeight Quantifier [0..1]	The maximum load by weight (including passengers) that the Mobility Capability can transport.

9.1.18 OperationalCapability

Type: Class
Package: PlanExecution
Operational capability describes the overall capabilities of a resource comprising its associated personnel and equipment. It accounts for training, readiness and equipment status.

Table 9.16 – Attributes of Class OperationalCapability

Attribute	Notes
level Descriptor	The organizational level at which the operational capability is intended to be performed.

9.1.19 Plan

Type: Class
Package: PlanExecution
A Plan represents an aggregated set of objectives and the resources and tasking to achieve them.

Table 9.17 – Attributes of Class Plan

Attribute	Notes
planType PlanType [0..1]	The extensible categorization of the type of plan.

9.1.20 PlanExecutionConstituent

Type: Class
Package: PlanExecution
An abstract class for constituent elements of tactical plan execution

Table 9.18 – Attributes of Class PlanExecutionConstituent

Attribute	Notes
«key» **id** ConstituentRef	The unique identifier for the instance
state PlanExecutionConstituentState	The state of the plan constituent according to the PlanExecutionConstituent state machine.
extendedStatus ExtendedPlanStatus	The extensible detailed categorization of the state of the plan execution constituent.
timeSpan Period [0..1]	The time during which the plan execution constituent is expected to be executed.
originator URI	The originator of the PlanExecutionConstituent. This is the component that instigated the creation of the instance.
progress Percentage [0..1]	The proportion of the plan execution constituent's objectives that have been achieved.

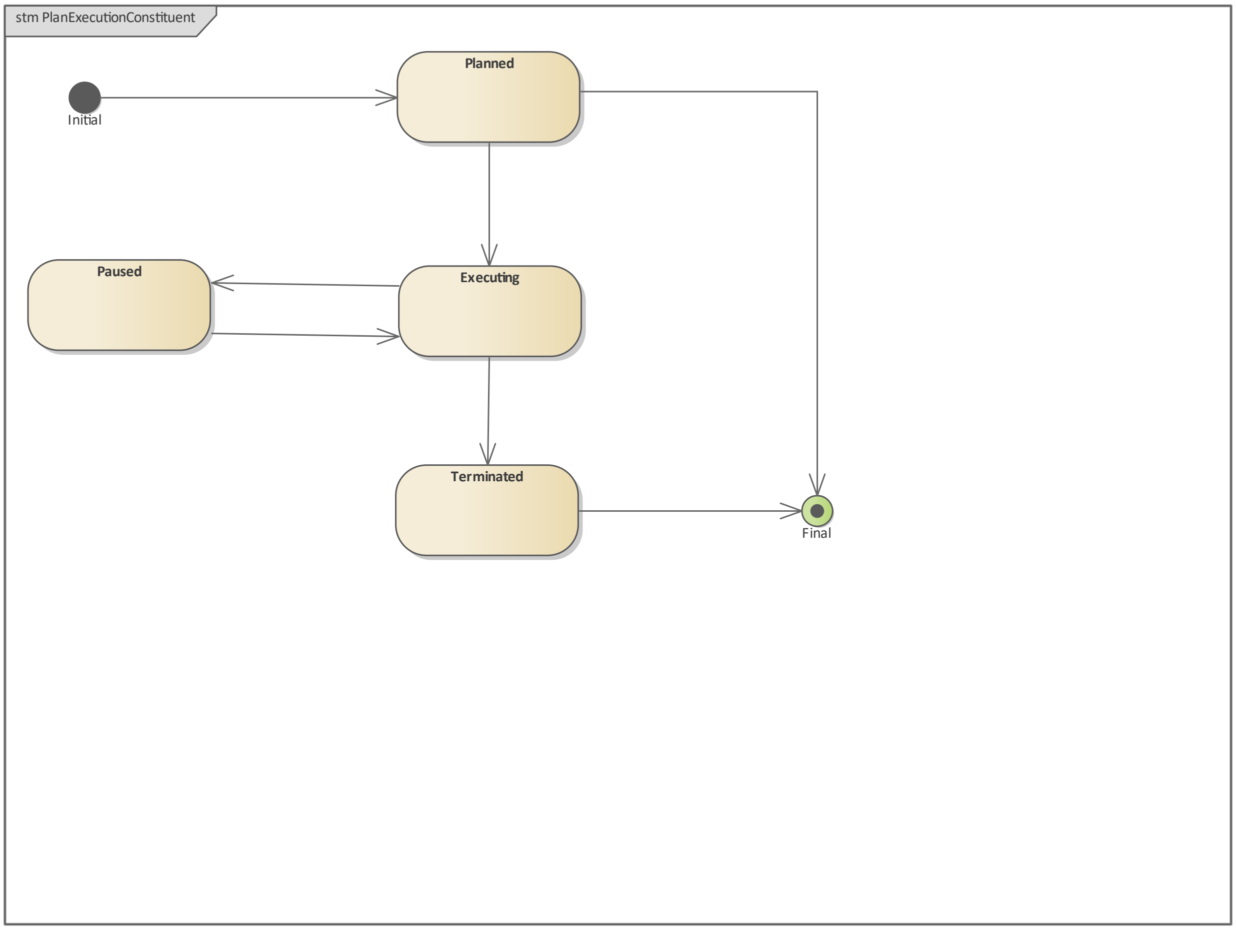


Figure 9.6 PlanExecutionConstituent (StateMachine diagram)

This diagram defines the state transitions of a plan execution constituent in response to recommendation actions from Tactical Decision Aids.

9.1.20.1 Executing

Type: State
Package: PlanExecution

A successful start or resume recommendation transition a plan constituent to this state. A plan constituent also transitions to this state at the time of the start of its time span.

9.1.20.2 Paused

Type: State
Package: PlanExecution

A successful pause recommendation transitions a plan constituent to this state. The behaviour of the Plan Execution component with respect to the end of a paused plan constituent's Time Span is implementation defined, but observable by Tactical Decision Aids through the relevant Data Sink interface.

9.1.20.3 Planned

Type: State
Package: PlanExecution

A successful plan recommendation creates a plan constituent in this state.

9.1.20.4 Terminated

Type: State
Package: PlanExecution
A successful terminate recommendation transitions a plan constituent to this state. Plan Execution should transition plan constituent instances that have been started to this state before they are deleted.

9.1.21 Resource

Type: Class
Package: PlanExecution
A Resource is an abstraction of a physical entity that can be independently tasked to achieve an objective.

Table 9.19 – Attributes of Class Resource

Attribute	Notes
«key» **id** ResourceRef	The unique identifier for the instance
weight Quantifier [0..1]	The current weight of the Resource
entity EntityRef [0..1]	A reference to the entity representing the resource in the tactical picture
extendedData AdditionalData	Additional dynamic information related to the Resource
readiness ReadinessDescriptor	The extensible categorization of the readiness of resource (the extent to which resource is ready and available to be tasked to employ its capabilities).
timeOfValidity DateTime	The time for which the attributes of the Resource class are valid.
derivation Derivation	A description of the means by which the data for the Resource's attributes were derived. This includes sensing, communication routes and human input.

9.1.22 ResourceMetaData

Type: Class
Package: PlanExecution

Table 9.20 – Attributes of Class ResourceMetaData

Attribute	Notes
distanceUnit DistanceUnit	The unit used to defined the Resources distance properties
speedUnit SpeedUnit	The unit used to define the Resource's speed properties
derivation Derivation	A description of the means by which the data for the Resource Property's attributes were derived. This includes sensing, communication routes and human input.

9.1.23 ResourceProperties

Type: Class
Package: PlanExecution
The static, persistent properties of the resource that are not expected to change as a plan is proposed and executed. Properties are specified in this data model that are expected to be particular pertinent to the planning of operational utilization of resources. For instance those that provide constraints on movement and demand conditions on the operating environment.

Table 9.21 – Attributes of Class ResourceProperties

Attribute	Notes
category ResourceCategory	The extensible categorization of the type of resource
extendedData AdditionalData	Additional static information related to the Resource
«key» **id** ResourceRef	The unique identifier for the instance

9.1.24 ResourceTasking

Type: Class
Package: PlanExecution
Resource Tasking is a Resource's contribution to a Task Objective

Table 9.22 – Attributes of Class ResourceTasking

Attribute	Notes
activity TaskingActivity	An extensible categorization of the activity that the resource has been tasked to undertake.

9.1.25 Space

Type: Class
Package: PlanExecution
A resource beyond the Earth's atmosphere

Table 9.23 – Attributes of Class Space

Attribute	Notes
orbit OrbitCategory	The kind of orbit or trajectory that the space resource is on

9.1.26 SubsurfaceVessel

Type: Class
Package: PlanExecution
A resource operating underwater

Table 9.24 – Attributes of Class SubsurfaceVessel

Attribute	Notes
maxOperatingDepth Distance [0..1]	The operating limit of the subsurface vehicle with respect to depth in the water

9.1.27 SurfaceVessel

Type: Class
Package: PlanExecution
A resource operating on water.

Table 9.25 – Attributes of Class SurfaceVessel

Attribute	Notes
maxSeaState Integer [0..1]	The maximum sea state in which the Surface Vessel can operate specified in terms of the World Meteorological Organization (WMO) sea state code - range 0 .. 9.

9.1.28 SurveillanceCapability

Type: Class
Package: PlanExecution
A capability to sense or observe objects in the operational environment.

Table 9.26 – Attributes of Class SurveillanceCapability

Attribute	Notes
reportingRate Quantifier [0..1]	
operatingBand Descriptor	A qualitative description of the band in which the Surveillance Capability operates
targetClassification ClassificationDescriptor [0..*]	The categories of object which the Surveillance Capability can detect
capacity Integer [0..1]	The number of objects that the Surveillance Capability can continuously monitor.

9.1.29 TargetCapability

Type: Class
Package: PlanExecution

Table 9.27 – Attributes of Class TargetCapability

Attribute	Notes
classification ClassificationDescriptor	A category of target for which the resource has a Fire Capability
successLikelihood Percentage	The nominal likelihood that an engagement with the specified target will be successful.

9.1.30 TaskObjective

Type: Class
Package: PlanExecution
A Task Objective represents the discrete intent with respect to a particular Entity from the tactical picture within the context of an overall Plan

Table 9.28 – Attributes of Class TaskObjective

Attribute	Notes
entityObject EntityRef [0..1]	The Entity from the tactical picture to which the intent of the Task Objective is directed.
category ObjectiveCategory	An extensible categorization of the kind of tasking objective set.
detail Detail	Extensible, additional system and/or domain specific description of the tasking objective
intent IntentDescriptor	Extensible categorization of the kind of effect intended with respect to the object of the tasking
priority Percentage [0..1]	Optional prioritization of task objectives. Higher percentages reflect higher priorities. The values for all objectives for a plan are not required to sum to 100%.

9.1.31 TransmissionCapability

Type: Class
Package: PlanExecution
A capability for the electronic transmission of data (including voice and video).

Table 9.29 – Attributes of Class TransmissionCapability

Attribute	Notes
power Quantifier [0..1]	The nominal output
dataRate Quantifier [0..1]	The rate at which the Transmission Capability can transmit data

Attribute	Notes
operatingBand Descriptor	A qualitative description of the band in which the Transmission Capability operates
protocol Descriptor [0..*]	The transmission protocols supported by the capability.
dataClassification Descriptor [0..*]	The classification of information supported by the capability

9.1.32 Vehicle

Type: Class
Package: PlanExecution
A resource with its own movement capabilities

Table 9.30 – Attributes of Class Vehicle

Attribute	Notes
maxSpeed Speed [0..1]	The maximum speed that at which the vehicle can move
cruisingSpeed Speed [0..1]	The optimum speed, for planning purposes, at which the vehicle transits between locations.
turnRate Quantifier [0..1]	The rate at which the vehicle can maneuver to change its heading within the horizontal plane (for planning purposes).

9.1.33 Vessel

Type: Class
Package: PlanExecution
A waterborne resource

Table 9.31 – Attributes of Class Vessel

Attribute	Notes
minOperatingDepth Distance [0..1]	The vessel resource's operating limit with respect to shallow water

9.1.34 AmmunitionCategory

Type: DataType
Package: PlanExecution
An abstraction of the categories of ammunition

9.1.35 CaliberCategory

Type: DataType
Package: PlanExecution
An abstraction of the categories of ammunition caliber.

9.1.36 CapabilityCategory

Type: DataType
Package: PlanExecution
An abstraction of the categories of capabilities

9.1.37 CapabilityRef

Type: DataType
Package: PlanExecution
A datatype with a platform specific mapping to represent a reference to a Capability. A reference is a unique identifier within the scope of the Plan Execution component implementing this specification.

9.1.38 ConstituentRef

Type: DataType
Package: PlanExecution
A reference to a Plan Execution Constituent. A reference is a unique identifier within the scope of the Plan Execution component implementing this specification.

9.1.39 DependencyCategory

Type: DataType
Package: PlanExecution
An abstraction of the categories of dependencies

9.1.40 DerivationDescriptor

Type: DataType
Package: PlanExecution
System specific description of the derivation of the associated data

9.1.41 ExtendedPlanStatus

Type: DataType
Package: PlanExecution
An abstraction of additional sub-categories of plan status; each sub-category logically maps to a specific plan state

9.1.42 IntentDescriptor

Type: DataType
Package: PlanExecution
An abstraction of the categories of intent.

9.1.43 ObjectiveCategory

Type: DataType
Package: PlanExecution
An abstraction of the categories of task objectives.

9.1.44 OrbitCategory

Type: DataType
Package: PlanExecution
An abstraction of the categories of orbits in space

9.1.45 PlanExecutionConstituentState

Type: Enumeration
Package: PlanExecution
Representation of the state machine for plan constituents.

Table 9.32 – Attributes of Enumeration PlanExecutionConstituentState

Attribute	Notes
«enum» **PLANNED**	The plan constituent has been created but is not yet being executed
«enum» **EXECUTING**	The plan constituent has been started but terminated and has been resumed after any pause.
«enum» **PAUSED**	The plan constituent has been paused, but not yet resumed after being executed.
«enum» **TERMINATED**	The plan constituent has been terminated after being executed.

9.1.46 PlanType

Type: DataType
Package: PlanExecution
An abstraction of the categories of plans.

9.1.47 ReadinessDescriptor

Type: DataType
Package: PlanExecution
An abstraction of the categories of readiness

9.1.48 ResourceCategory

Type: DataType
Package: PlanExecution
An abstraction of the categories of resources.

9.1.49 ResourceRef

Type: DataType
Package: PlanExecution
A datatype with a platform specific mapping to represent a reference to a Resource. A reference is a unique identifier within the scope of the Plan Execution component implementing this specification.

9.1.50 SpecificationDescriptor

Type: DataType
Package: PlanExecution
An abstraction of the categories of specifications

9.1.51 TaskingActivity

Type: DataType
Package: PlanExecution
An abstraction of the categories of tasking activity a resource can undertake.

9.2 TacticalPicture

Parent Package: DataModel
The Tactical Picture package in the Data Model describes the particular usage of the TACSIT Data Exchange (TEX) standard that satisfies this standard's tactical picture requirements.

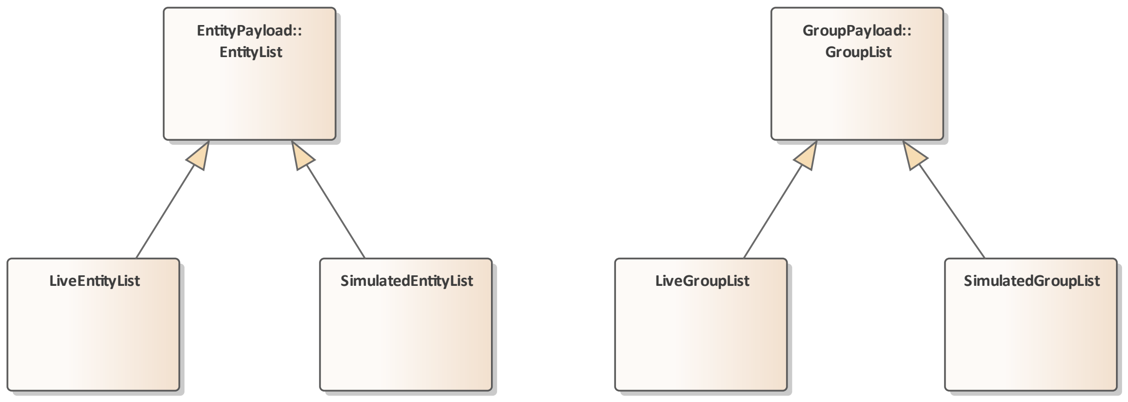


Figure 9.7 Live Simulated (Class diagram)

This diagram shows how the live and simulated versions of the tactical picture are represented using classes from the TACSIT Data Exchange (TEX) specification.

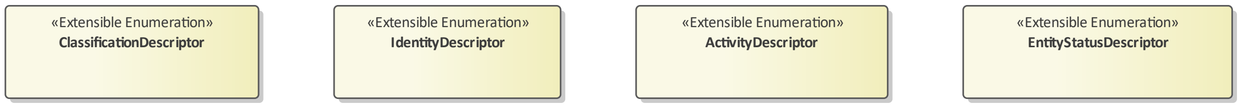


Figure 9.8 Track Categorization (Class diagram)

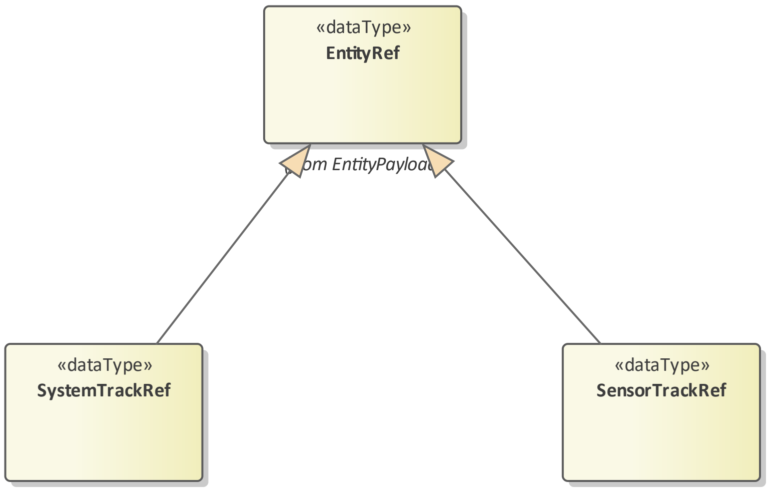


Figure 9.9 Tracks (Class diagram)

This diagram shows how system tracks and sensor tracks are represented using a reference to the Entity class from the TACSIT Data Exchange (TEX) specification.

9.2.1 LiveEntityList

Type: Class
Package: TacticalPicture
The list of entities contributing to the live tactical picture (i.e. relating to the real operational environment)

9.2.2 LiveGroupList

Type: Class
Package: TacticalPicture
The list of groups contributing to the live tactical picture (i.e. relating to the real operational environment)

9.2.3 SimulatedEntityList

Type: Class
Package: TacticalPicture
The list of entities contributing to the simulated tactical picture (i.e. relating to a simulation of the operational environment)

9.2.4 SimulatedGroupList

Type: Class
Package: TacticalPicture
The list of groups contributing to the simulated tactical picture (i.e. relating to a simulation of the operational environment)

9.2.5 ActivityDescriptor

Type: DataType
Package: TacticalPicture
Extensible definition of a track's activity. This class has an implementation specific null value that has the meaning of no statement being made in regard of the category of activity.

9.2.6 ClassificationDescriptor

Type: DataType
Package: TacticalPicture
Extensible definition of a track's classification. This class has an implementation specific null value that has the meaning of no statement being made in regard of the category of classification.

9.2.7 EntityStatusDescriptor

Type: DataType
Package: TacticalPicture
Extensible definition of an enitity's status.

9.2.8 IdentityDescriptor

Type: DataType
Package: TacticalPicture
Extensible definition of a track's identity. This class has an implementation specific null value that has the meaning of no statement being made in regard of the category of identity.

9.2.9 SensorTrackRef

Type: DataType
Package: TacticalPicture
A reference to a sensor track - i.e. a track object from the perspective of a sensor subsystem

9.2.10 SystemTrackRef

Type: DataType
Package: TacticalPicture
A reference to a system track - i.e. a track object from the perspective of the compiled tactical picture of a C2 (Command and Control) system.

This page intentionally left blank.

10 ServiceModel

Parent Package: tactical decision aids
The Tactical Decision Aids Service Model defines the operations that enable the flow of information from a Picture Management and a Plan Execution component to Tactical Decision Aid Components as well as the receipt of recommendations from Tactical Decision Aid components by the Tactical Picture and Plan Execution components.
The connection between components is initiated by the Tactical Decision Aid components using a PSM method. These components may require security permissions to do, in which case these are authenticated by a PSM protocol.
Use of a Data Sink Listener to subscribe to a series of change events requires a long-lived connection between the Tactical Decision Aid and the Tactical Picture or Plan Execution component providing the Data Sink interface. Other interface operations are self-contained requests initiated by the Tactical Decision Aid component and do not require long-lived connections.
Tactical Decision Aids components make recommendations based on their own internal business logic, information they have access to by other means (including input from system users) and information received from the Tactical Picture and Plan Execution components through the Data Sink interfaces.

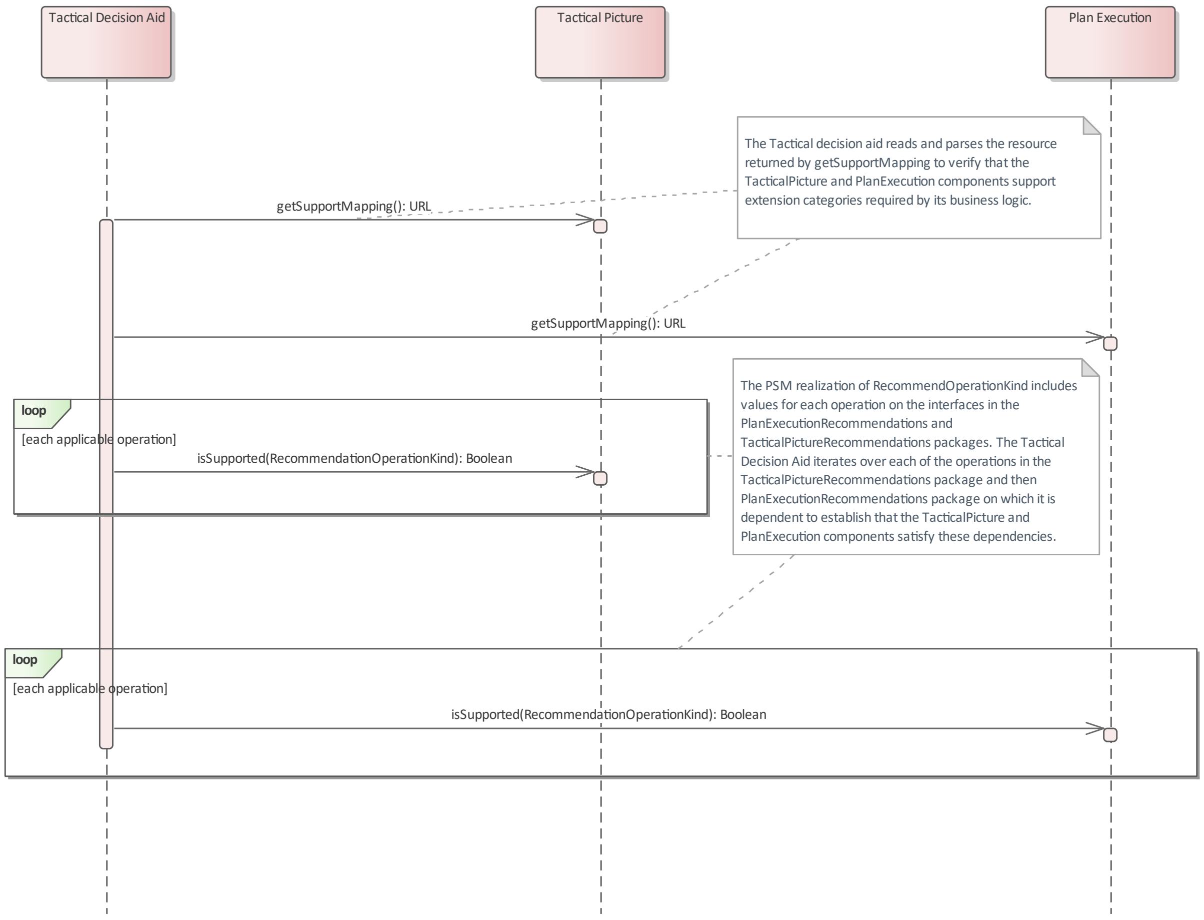
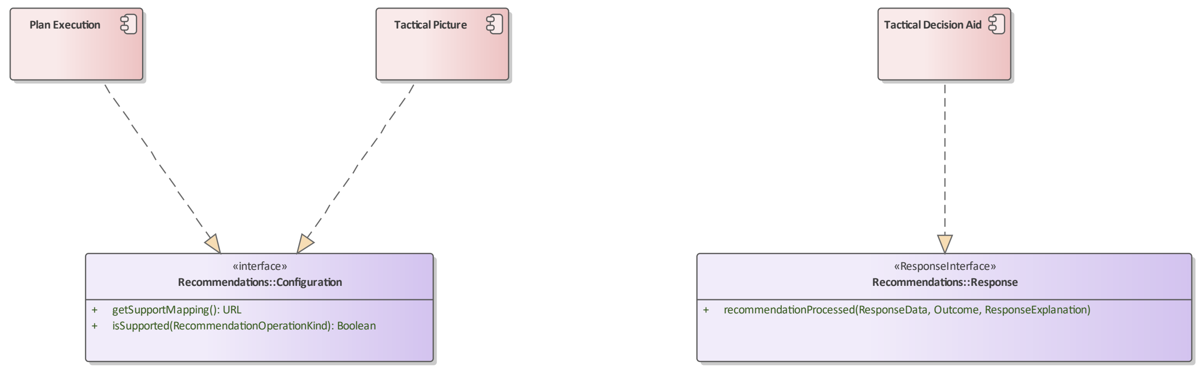
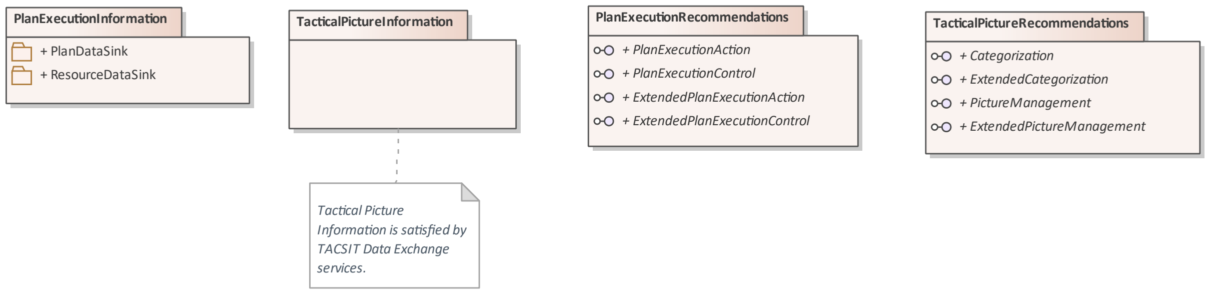

Tactical Decision Aid
Tactical Picture
Plan Execution
The Tactical decision aid reads and parses the resource returned by getSupportMapping to verify that the TacticalPicture and PlanExecution components support extension categories required by its business logic.
getSupportMapping(): URL
getSupportMapping(): URL
loop
[each applicable operation]
isSupported(RecommendationOperationKind): Boolean
The PSM realization of RecommendOperationKind includes values for each operation on the interfaces in the PlanExecutionRecommendations and TacticalPictureRecommendations packages. The Tactical Decision Aid iterates over each of the operations in the TacticalPictureRecommendations package and then PlanExecutionRecommendations package on which it is dependent to establish that the TacticalPicture and PlanExecution components satisfy these dependencies.
loop
[each applicable operation]
isSupported(RecommendationOperationKind): Boolean

Figure 10.1 Recommendations (Interaction diagram)



Plan Execution
Tactical Picture
Tactical Decision Aid
«interface»
Recommendations::Configuration
+ getSupportMapping(): URL
+ isSupported(RecommendationOperationKind): Boolean
«ResponseInterface»
Recommendations::Response
+ recommendationProcessed(ResponseData, Outcome, ResponseExplanation)

Figure 10.2 Recommendations Service Mapping (Component diagram)



PlanExecutionInformation
+ PlanDataSink
+ ResourceDataSink
TacticalPictureInformation
Tactical Picture Information is satisfied by TACSIT Data Exchange services.
PlanExecutionRecommendations
+ PlanExecutionAction
+ PlanExecutionControl
+ ExtendedPlanExecutionAction
+ ExtendedPlanExecutionControl
TacticalPictureRecommendations
+ Categorization
+ ExtendedCategorization
+ PictureManagement
+ ExtendedPictureManagement

Figure 10.3 ServiceModel (Package diagram)

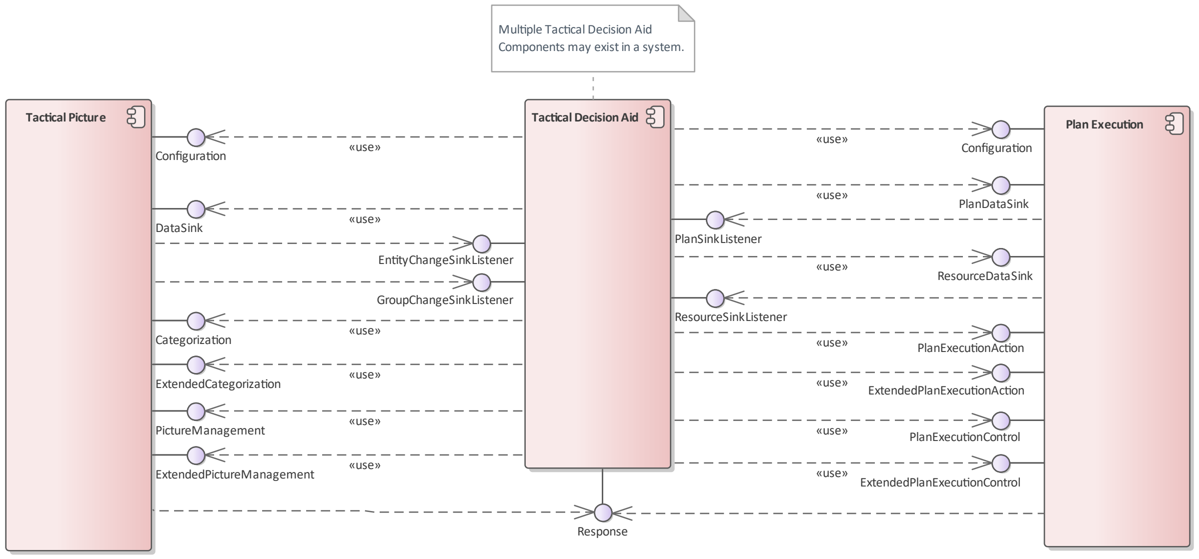


Figure 10.4 ServiceModel (Component diagram)

10.1 Plan Execution

Type: Component
Package: ServiceModel
Abstract component representing components with the functionality to manage and monitor the status of plans as they are executed. Plan Execution components receive information from system users, tactical data-links, databases and other components through interfaces outside of the scope of this specification. Tactical Decision Aids receive information about all plans; the plans they have initiated, plans from other Tactical Decision Aids and plans originating from outside the scope of this specification.

10.2 Tactical Decision Aid

Type: Component
Package: ServiceModel
Abstract component representing components that provide the functionality to assist with the making of tactical decisions

10.3 Tactical Picture

Type: Component
Package: ServiceModel
Abstract component representing components that provide the functionality of compiling and managing the tactical picture. Tactical Picture components receive information from sensors, system users and other components through interfaces outside of the scope of this specification.

10.4 PlanExecutionInformation

Parent Package: ServiceModel
The interfaces to allow Tactical Decision Aids to receive Plan Execution Information. This is achieved through two instances of the Data Sink pattern. One enables Tactical Decision Aids to receive a current view and then changes to Plan Execution Constituents (Plans and their sub-components Task Objectives and Resource Taskings). The other provides an equivalent service for Resources and their composite Capabilities and Dependencies.
Navigability of associations between classes in the Plan Execution data model is facilitated by id attributes with a key stereotype. Navigation of the associations between objects delivered by the Data Sink services is achieved by a PSM specific methods using the id key attributes.

10.4.1 PlanDataSink

Parent Package: PlanExecutionInformation
The interfaces to allow Tactical Decision Aids to receive Plan Information

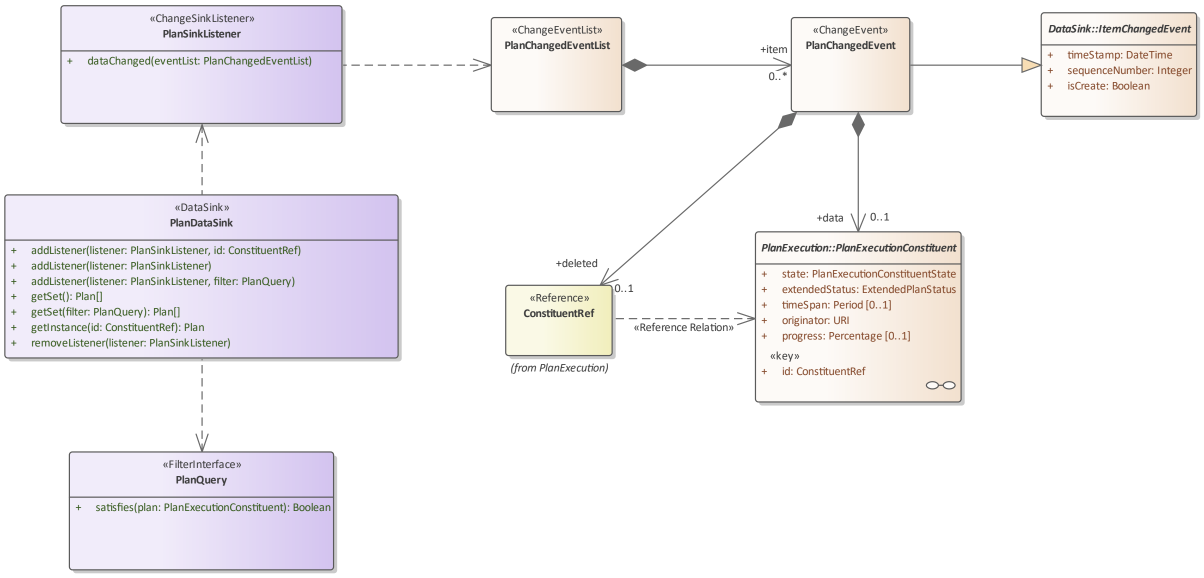

«ChangeSinkListener»
PlanSinkListener
+ dataChanged(eventList: PlanChangedEventList)
«ChangeEventList»
PlanChangedEventList
+item
0..*
«ChangeEvent»
PlanChangedEvent
DataSink::ItemChangedEvent
+ timeStamp: DateTime
+ sequenceNumber: Integer
+ isCreate: Boolean
«DataSink»
PlanDataSink
+ addListener(listener: PlanSinkListener, id: ConstituentRef)
+ addListener(listener: PlanSinkListener)
+ addListener(listener: PlanSinkListener, filter: PlanQuery)
+ getSet(): Plan[]
+ getSet(filter: PlanQuery): Plan[]
+ getInstance(id: ConstituentRef): Plan
+ removeListener(listener: PlanSinkListener)
+deleted
0..1
+data
0..1
«Reference»
ConstituentRef
(from PlanExecution)
«Reference Relation»
PlanExecution::PlanExecutionConstituent
+ state: PlanExecutionConstituentState
+ extendedStatus: ExtendedPlanStatus
+ timeSpan: Period [0..1]
+ originator: URI
+ progress: Percentage [0..1]
«key»
+ id: ConstituentRef
«FilterInterface»
PlanQuery
+ satisfies(plan: PlanExecutionConstituent): Boolean

Figure 10.5 PlanDataSink (Class diagram)

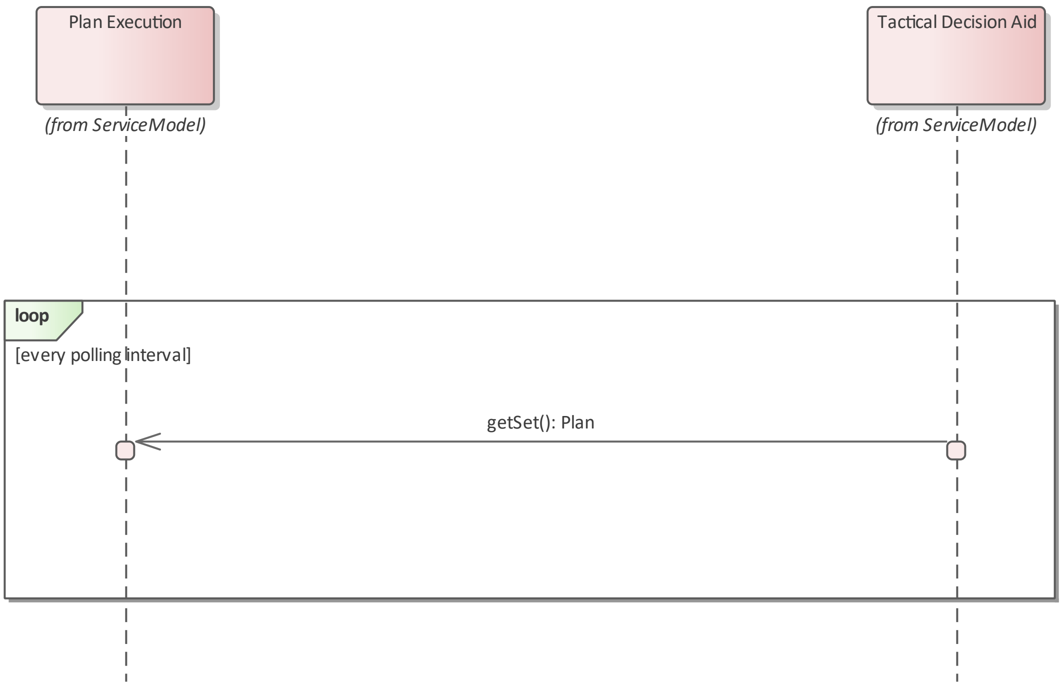


Figure 10.6 PlanDataSink – All Plan – Polling (Interaction diagram)

Use of the PlanDataSink interface to get a regular view of all plans. To receive a subset of plans the getSet with a PlanQuery parameter operation is used.

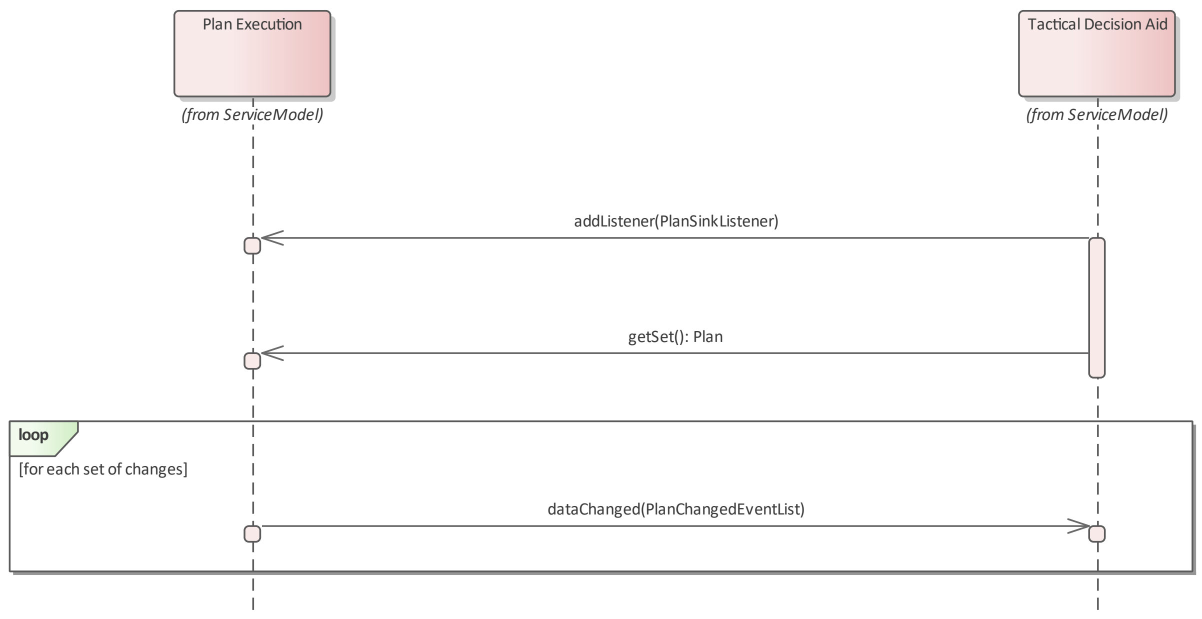


Figure 10.7 PlanDataSink – All Plans – On Change (Interaction diagram)

Use of the PlanDataSink interface to get an initial view of all plans and then receive updates on changes for an on-change style of use of the interface. The listener interface is added first so that events are not missed. In this scenario it is preferable to process no-change events than to miss events.

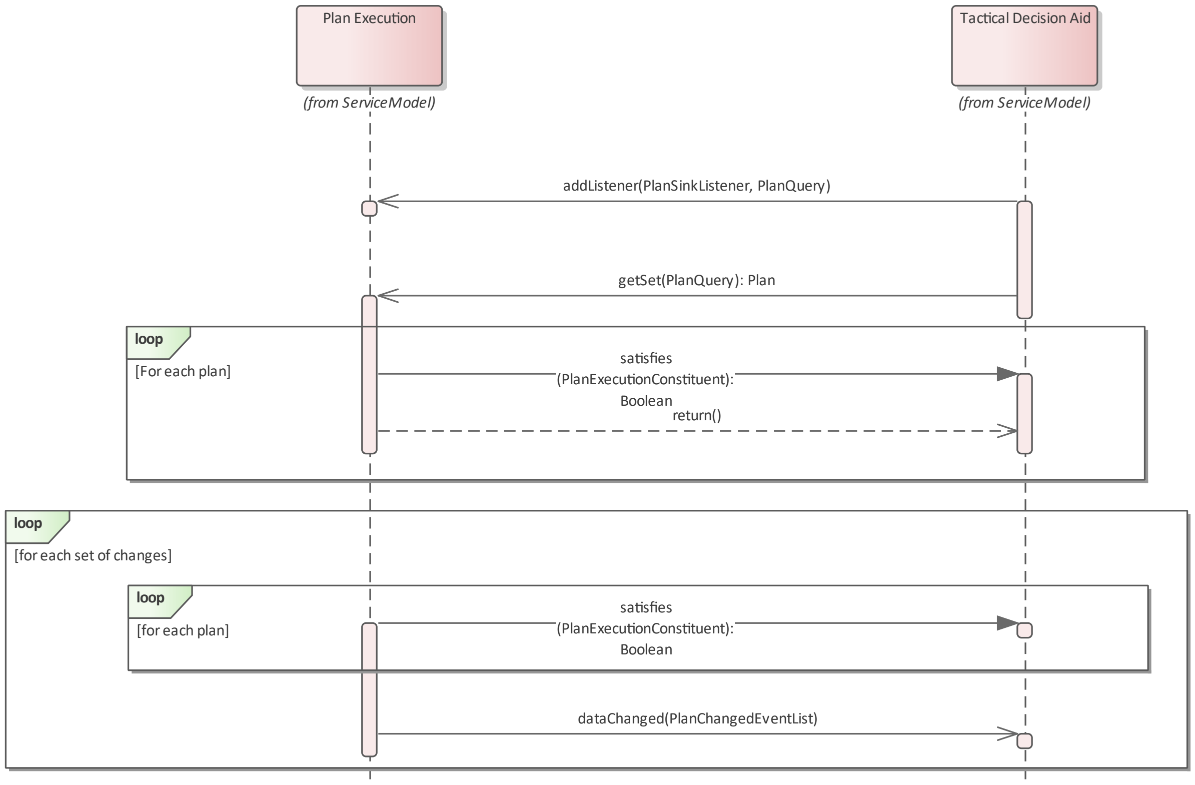

Plan Execution
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
addListener(PlanSinkListener, PlanQuery)
getSet(PlanQuery): Plan
loop
[For each plan]
satisfies (PlanExecutionConstituent): Boolean
return()
loop
[for each set of changes]
loop
[for each plan]
satisfies (PlanExecutionConstituent): Boolean
dataChanged(PlanChangedEventList)

Figure 10.8 PlanDataSink – Filtered Plans (Interaction diagram)

Use of the PlanDataSink interface to get an initial view of a subset of plans and then receive updates on changes.

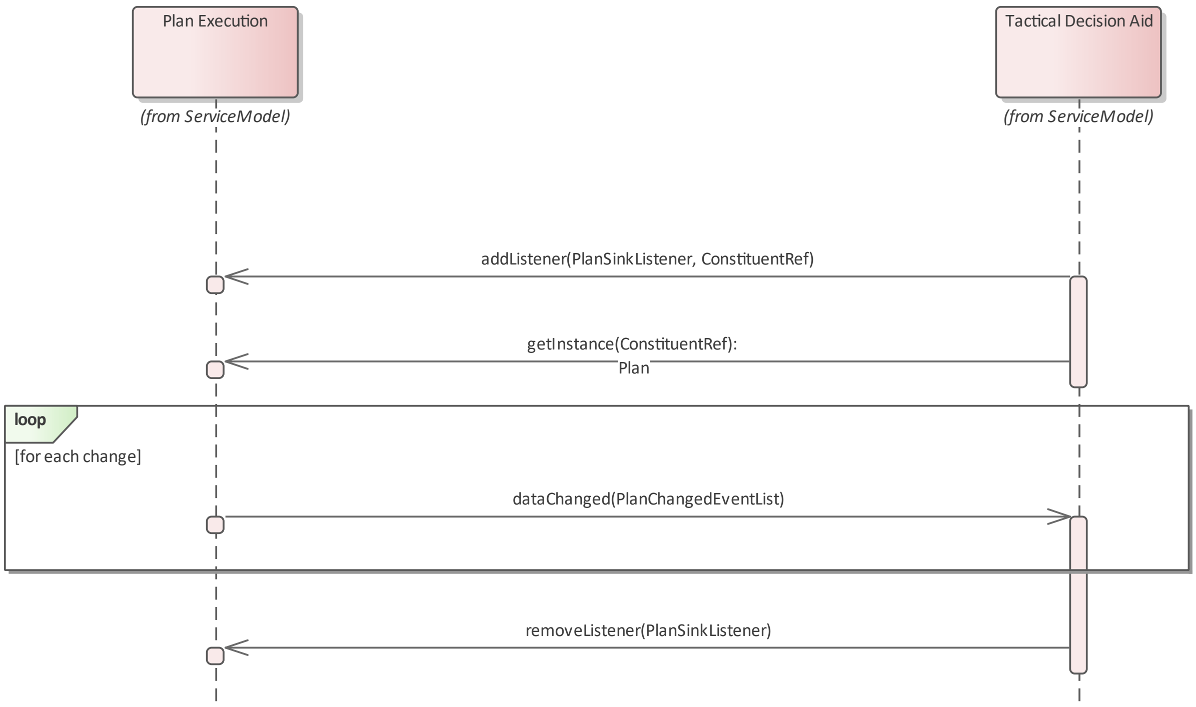

Plan Execution
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
addListener(PlanSinkListener, ConstituentRef)
getInstance(ConstituentRef): Plan
loop
[for each change]
dataChanged(PlanChangedEventList)
removeListener(PlanSinkListener)

Figure 10.9 PlanDataSink – Single Plan (Interaction diagram)

Use of the PlanDataSink interface to get an initial view of a specific plan and then receive updates on changes.

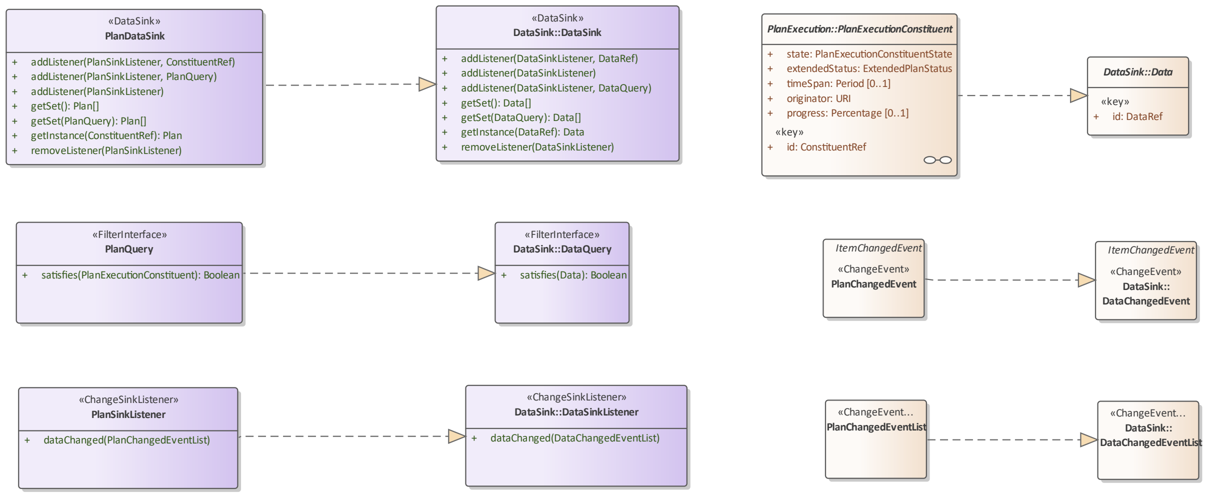

«DataSink»
PlanDataSink
+ addListener(PlanSinkListener, ConstituentRef)
+ addListener(PlanSinkListener, PlanQuery)
+ addListener(PlanSinkListener)
+ getSet(): Plan[]
+ getSet(PlanQuery): Plan[]
+ getInstance(ConstituentRef): Plan
+ removeListener(PlanSinkListener)
«DataSink»
DataSink::DataSink
+ addListener(DataSinkListener, DataRef)
+ addListener(DataSinkListener)
+ addListener(DataSinkListener, DataQuery)
+ getSet(): Data[]
+ getSet(DataQuery): Data[]
+ getInstance(DataRef): Data
+ removeListener(DataSinkListener)
PlanExecution::PlanExecutionConstituent
+ state: PlanExecutionConstituentState
+ extendedStatus: ExtendedPlanStatus
+ timeSpan: Period [0..1]
+ originator: URI
+ progress: Percentage [0..1]
«key»
+ id: ConstituentRef
DataSink::Data
«key»
+ id: DataRef
«FilterInterface»
PlanQuery
+ satisfies(PlanExecutionConstituent): Boolean
«FilterInterface»
DataSink::DataQuery
+ satisfies(Data): Boolean
ItemChangedEvent
«ChangeEvent»
PlanChangedEvent
ItemChangedEvent
«ChangeEvent»
DataSink::
DataChangedEvent
«ChangeSinkListener»
PlanSinkListener
+ dataChanged(PlanChangedEventList)
«ChangeSinkListener»
DataSink::DataSinkListener
+ dataChanged(DataChangedEventList)
«ChangeEvent...
PlanChangedEventList
«ChangeEvent...
DataSink::
DataChangedEventList

Figure 10.10 PlanDataSink Realization (Class diagram)

10.4.1.1 PlanChangedEvent

Type: Class
Package: PlanDataSink
Represents information about a change to a Plan

10.4.1.2 PlanChangedEventList

Type: Class
Package: PlanDataSink
Represents the list of changes to Plans since the last event notified to that instance of the listener. Multiple changes may be consolidated into a single callback to a listener on the interface.

10.4.1.3 PlanDataSink

Type: Interface
Package: PlanDataSink
This interface contains operations that give a Tactical Decision Aid access to information about the execution of plan constituents. A Tactical Decision Aid can add and remove listeners as well as reading the information about individual plan constituents or all or a filtered subset of plan constituents.

Table 10.1 – Methods of Interface PlanDataSink

Method	**Notes**	**Parameters**
addListener()	Operation to add a listener for callbacks relating to a single plan constituent	PlanSinkListener **listener** The listener object to receive the callback ConstituentRef **id** A reference to the specific plan instance of interest
addListener()	Operation to add a listener for callbacks relating to all plan constituents that satisfy the Query (including plans created or meeting the query subsequently)	PlanSinkListener **listener** The listener object to receive the callback PlanQuery **filter** The object to filter changes to plans
addListener()	Operation to add a listener for callbacks relating to all plan constituents, including plan constituents subsequently created.	PlanSinkListener **listener** The listener object to receive the callback
getSet()	Operation to obtain the information relating to all plan constituents	
getSet()	Operation to obtain the information relating to all the plan constituents satisfying the query	PlanQuery **filter** The object to filter plan instances
getInstance()	Operation to obtain the information relating to the plan constituent reference	ConstituentRef **id** A reference to the specific plan instance of interest
removeListener()	Operation to remove a listener	PlanSinkListener **listener** The listener object to no longer receive callbacks

10.4.1.4 PlanQuery

Type: Interface
Package: PlanDataSink
This is an interface through which a client can define Queries on plan constituents so as to filter the information returned. Classes implementing the interface provide means to set the query parameters (such as a constructor).

Table 10.2 – Methods of Interface PlanQuery

Method	Notes	Parameters
satisfies()	This operation is the client's implementation of a filtering query for plan constituents	PlanExecutionConstituent **plan** The plan to which to apply the filter

10.4.1.5 PlanSinkListener

Type: Interface
Package: PlanDataSink
This is an interface for clients to implement callback to receive information on changes to plan constituents.

Table 10.3 – Methods of Interface PlanSinkListener

Method	Notes	Parameters
dataChanged()	This operation is implemented by the client to process the data changed callback. Multiple changes can be notified through a single invocation.	PlanChangedEventList **eventList** The list of plan changes recevied by the listener

10.4.2 ResourceDataSink

Parent Package: PlanExecutionInformation
The interfaces to allow Tactical Decision Aids to receive Resource Information.

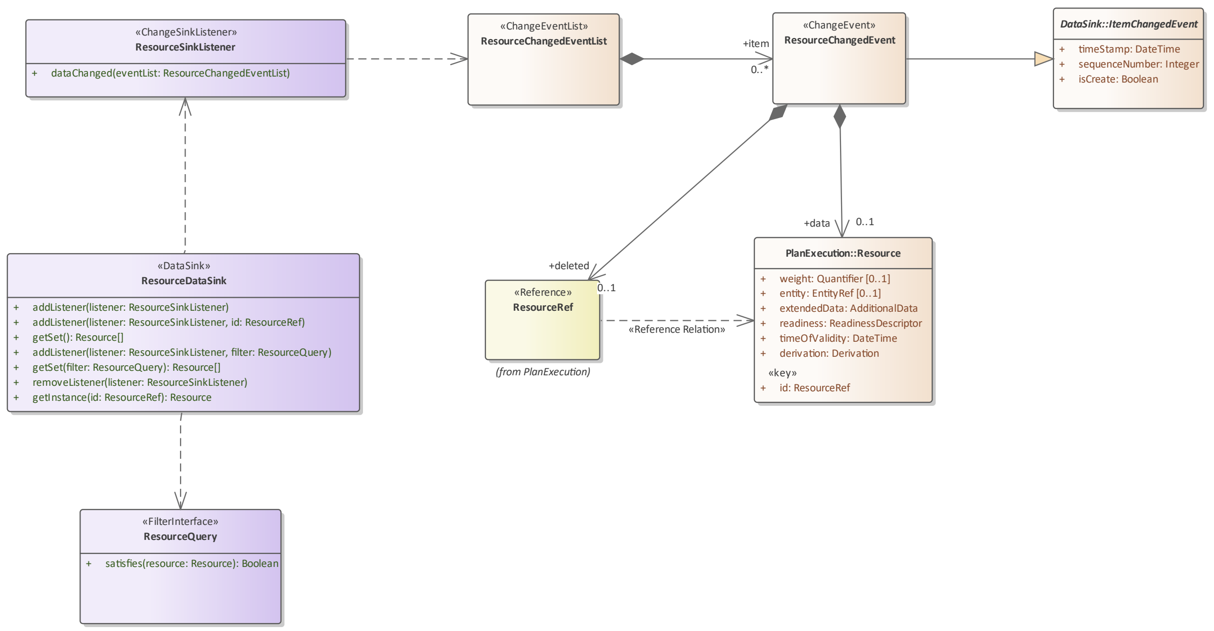

«ChangeSinkListener»
ResourceSinkListener
+ dataChanged(eventList: ResourceChangedEventList)
«ChangeEventList»
ResourceChangedEventList
+item
0..*
«ChangeEvent»
ResourceChangedEvent
DataSink::ItemChangedEvent
+ timeStamp: DateTime
+ sequenceNumber: Integer
+ isCreate: Boolean
+data
0..1
+deleted
0..1
«Reference»
ResourceRef
(from PlanExecution)
«Reference Relation»
PlanExecution::Resource
+ weight: Quantifier [0..1]
+ entity: EntityRef [0..1]
+ extendedData: AdditionalData
+ readiness: ReadinessDescriptor
+ timeOfValidity: DateTime
+ derivation: Derivation
«key»
+ id: ResourceRef
«DataSink»
ResourceDataSink
+ addListener(listener: ResourceSinkListener)
+ addListener(listener: ResourceSinkListener, id: ResourceRef)
+ getSet(): Resource[]
+ addListener(listener: ResourceSinkListener, filter: ResourceQuery)
+ getSet(filter: ResourceQuery): Resource[]
+ removeListener(listener: ResourceSinkListener)
+ getInstance(id: ResourceRef): Resource
«FilterInterface»
ResourceQuery
+ satisfies(resource: Resource): Boolean

Figure 10.11 ResourceDataSink (Class diagram)

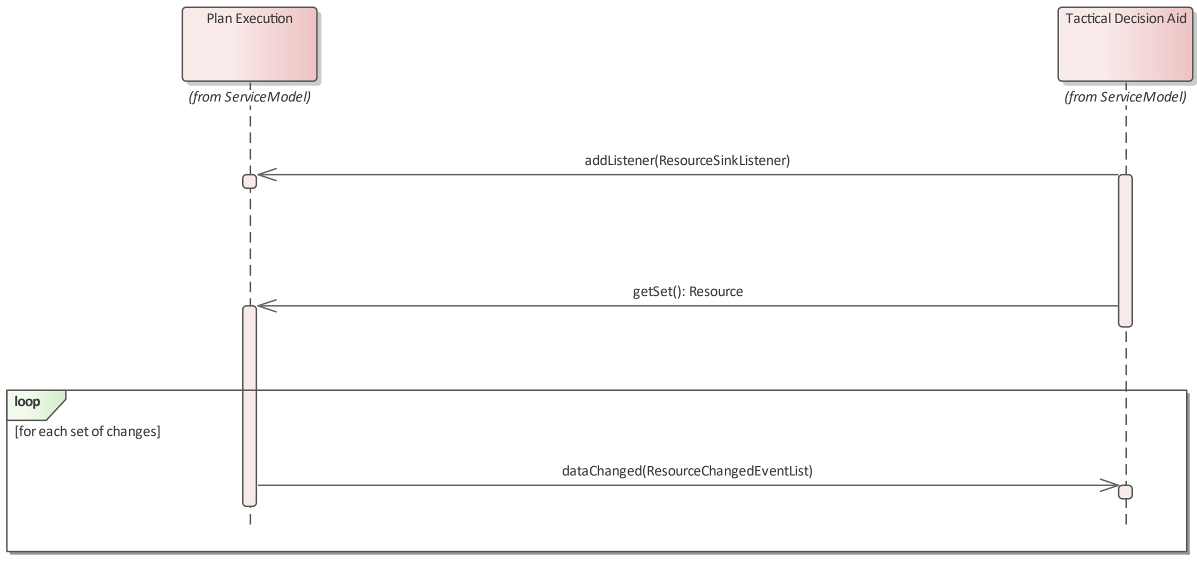

Plan Execution
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
addListener(ResourceSinkListener)
getSet(): Resource
loop
[for each set of changes]
dataChanged(ResourceChangedEventList)

Figure 10.12 ResourceDataSink – All Resources – On Change (Interaction diagram)

Use of the ResourceDataSink interface to get an initial view of all resources and then receive updates on changes for an on-change style of use of the interface. The listener interface is added first so that events are not missed. In this scenario it is preferable to process no-change events than to miss events.

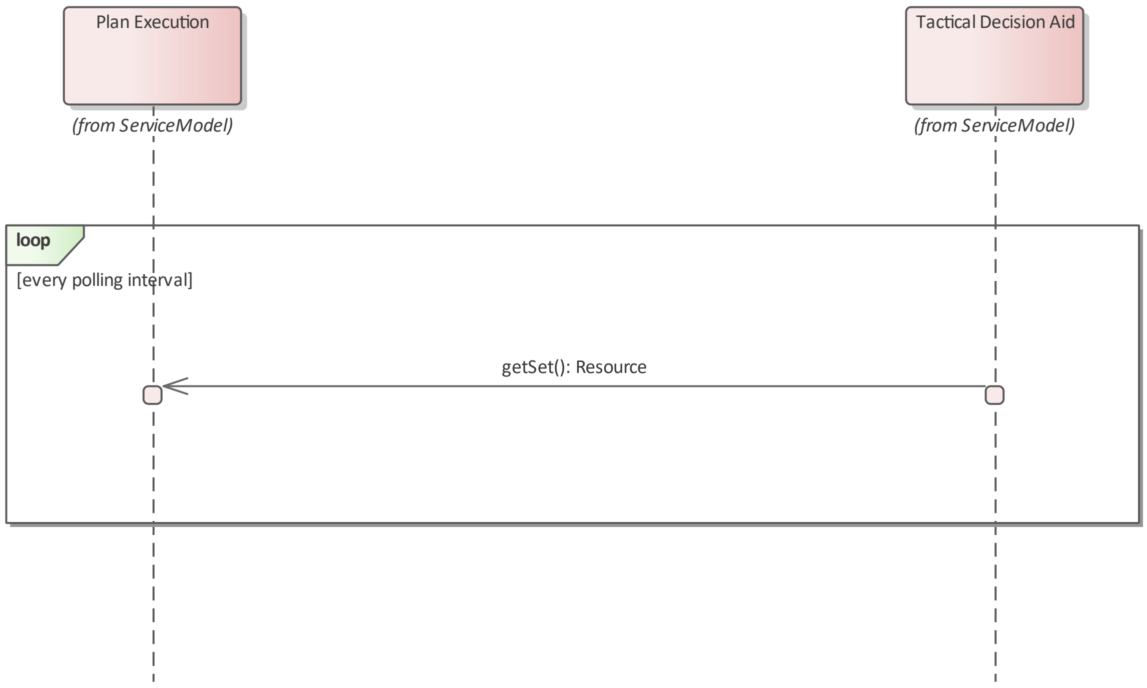
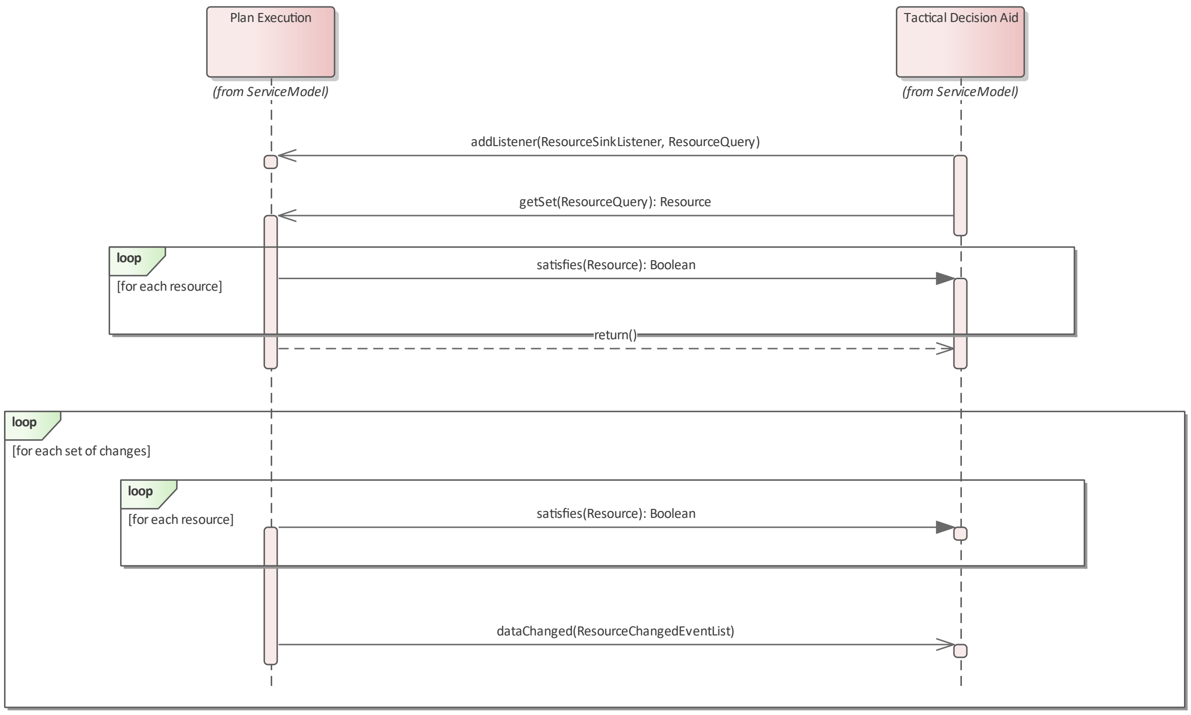


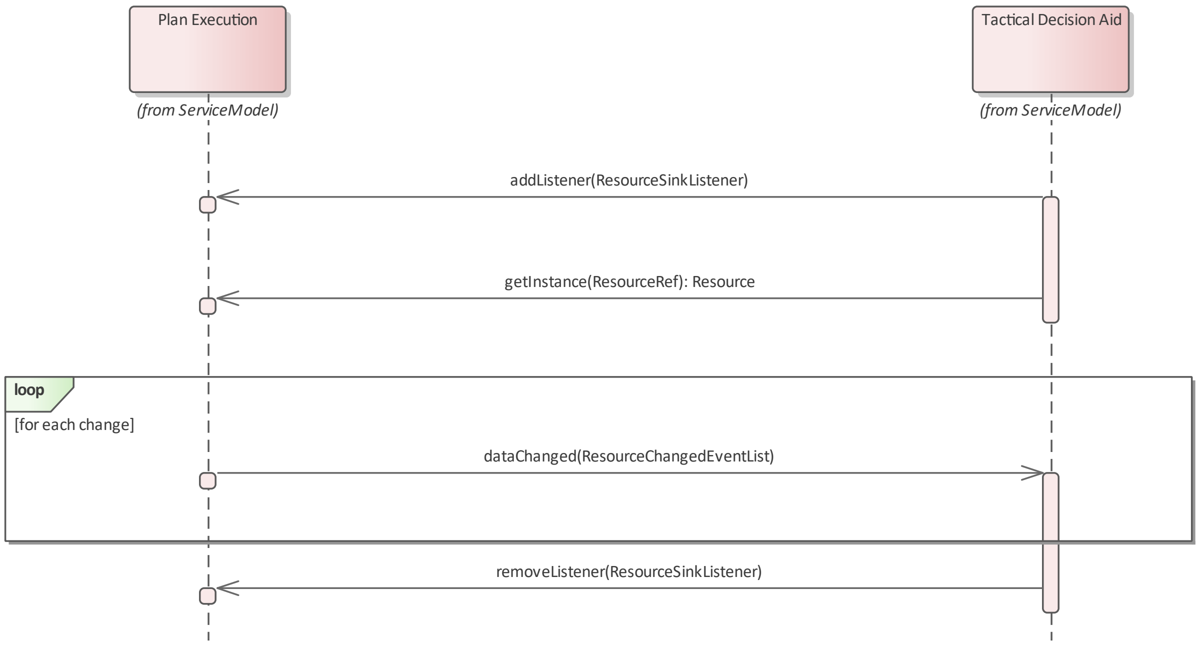
Figure 10.13 ResourceDataSink – All Resources – Polling (Interaction diagram)



Plan Execution
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
addListener(ResourceSinkListener, ResourceQuery)
getSet(ResourceQuery): Resource
loop
[for each resource]
satisfies(Resource): Boolean
return()
loop
[for each set of changes]
loop
[for each resource]
satisfies(Resource): Boolean
dataChanged(ResourceChangedEventList)

Figure 10.14 ResourceDataSink – Filtered Resources (Interaction diagram)

Use of the ResourceDataSink interface to get an initial view of a subset of resources and then receive updates on changes.



Plan Execution
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
addListener(ResourceSinkListener)
getInstance(ResourceRef): Resource
loop
[for each change]
dataChanged(ResourceChangedEventList)
removeListener(ResourceSinkListener)

Figure 10.15 ResourceDataSink – Single Resource (Interaction diagram)

Use of the ResourceDataSink interface to get an initial view of a specific resource and then receive updates on changes.

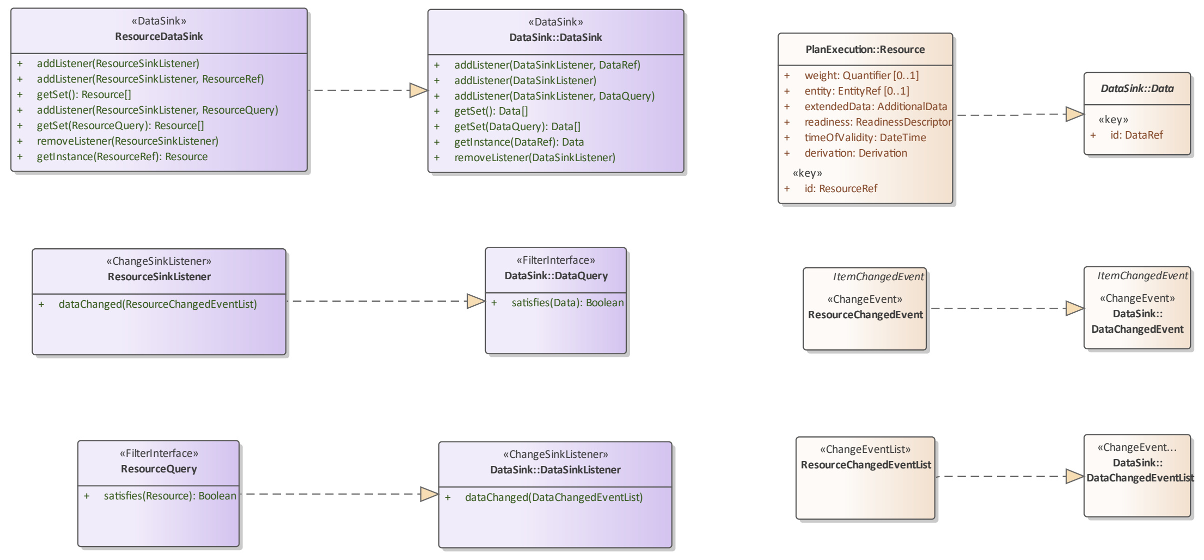

«DataSink»
ResourceDataSink
+ addListener(ResourceSinkListener)
+ addListener(ResourceSinkListener, ResourceRef)
+ getSet(): Resource[]
+ addListener(ResourceSinkListener, ResourceQuery)
+ getSet(ResourceQuery): Resource[]
+ removeListener(ResourceSinkListener)
+ getInstance(ResourceRef): Resource
«DataSink»
DataSink::DataSink
+ addListener(DataSinkListener, DataRef)
+ addListener(DataSinkListener)
+ addListener(DataSinkListener, DataQuery)
+ getSet(): Data[]
+ getSet(DataQuery): Data[]
+ getInstance(DataRef): Data
+ removeListener(DataSinkListener)
PlanExecution::Resource
+ weight: Quantifier [0..1]
+ entity: EntityRef [0..1]
+ extendedData: AdditionalData
+ readiness: ReadinessDescriptor
+ timeOfValidity: DateTime
+ derivation: Derivation
«key»
+ id: ResourceRef
DataSink::Data
«key»
+ id: DataRef
«ChangeSinkListener»
ResourceSinkListener
+ dataChanged(ResourceChangedEventList)
«FilterInterface»
DataSink::DataQuery
+ satisfies(Data): Boolean
ItemChangedEvent
«ChangeEvent»
ResourceChangedEvent
ItemChangedEvent
«ChangeEvent»
DataSink::
DataChangedEvent
«FilterInterface»
ResourceQuery
+ satisfies(Resource): Boolean
«ChangeSinkListener»
DataSink::DataSinkListener
+ dataChanged(DataChangedEventList)
«ChangeEventList»
ResourceChangedEventList
«ChangeEvent...
DataSink::
DataChangedEventList

Figure 10.16 ResourceDataSink Realization (Class diagram)

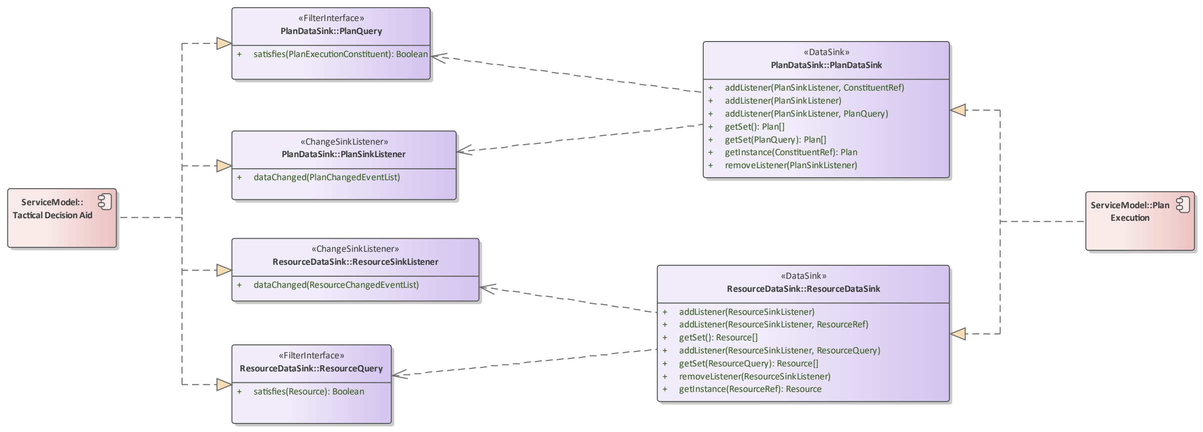

ServiceModel::
Tactical Decision Aid
«FilterInterface»
PlanDataSink::PlanQuery
+ satisfies(PlanExecutionConstituent): Boolean
«ChangeSinkListener»
PlanDataSink::PlanSinkListener
+ dataChanged(PlanChangedEventList)
«ChangeSinkListener»
ResourceDataSink::ResourceSinkListener
+ dataChanged(ResourceChangedEventList)
«FilterInterface»
ResourceDataSink::ResourceQuery
+ satisfies(Resource): Boolean
«DataSink»
PlanDataSink::PlanDataSink
+ addListener(PlanSinkListener, ConstituentRef)
+ addListener(PlanSinkListener)
+ addListener(PlanSinkListener, PlanQuery)
+ getSet(): Plan[]
+ getSet(PlanQuery): Plan[]
+ getInstance(ConstituentRef): Plan
+ removeListener(PlanSinkListener)
«DataSink»
ResourceDataSink::ResourceDataSink
+ addListener(ResourceSinkListener)
+ addListener(ResourceSinkListener, ResourceRef)
+ getSet(): Resource[]
+ addListener(ResourceSinkListener, ResourceQuery)
+ getSet(ResourceQuery): Resource[]
+ removeListener(ResourceSinkListener)
+ getInstance(ResourceRef): Resource
ServiceModel::Plan
Execution

Figure 10.17 PlanExecutionInformation Service Mapping (Component diagram)

10.4.2.1 ResourceChangedEvent

Type: Class
Package: ResourceDataSink
Represents information about a change to a Resource.

10.4.2.2 ResourceChangedEventList

Type: Class
Package: ResourceDataSink
Represents the list of changes to Resources since the last event notified to that instance of the listener. Multiple changes may be consolidated into a single callback to a listener on the interface.

10.4.2.3 ResourceDataSink

Type: Interface
Package: ResourceDataSink
This interface contains operations that give a Tactical Decision Aid access to information about resources that can execute plans. A Tactical Decision Aid can add and remove listeners as well as reading the information about individual resources or all or a filtered subset of resources.

Table 10.4 – Methods of Interface ResourceDataSink

Method	Notes	Parameters
addListener()	Operation to add a listener for callbacks relating to a single Resource (including Resources created subsequently)	ResourceSinkListener **listener** The listener object to receive the callback
addListener()	Operation to add a listener for callbacks relating to all Resources	ResourceSinkListener **listener** The listener object to receive the callback ResourceRef **id** A reference to the specific resource instance of interest
getSet()	Operation to obtain the information relating to all the Resources	
addListener()	Operation to add a listener for callbacks relating to all Resources that satisfy the Query (including Resources created or meeting the filter subsequently)	ResourceSinkListener **listener** The listener object to receive the callback ResourceQuery **filter** The object to filter changes to resources
getSet()	Operation to obtain the information relating to all the Resources satisfying the query	ResourceQuery **filter** The object to filter resource instances
removeListener()	Operation to remove a listener	ResourceSinkListener **listener** The listener object to no longer receive callbacks
getInstance()	Operation to obtain the information relating to the Resource reference	ResourceRef **id** A reference to the specific resource instance of interest

10.4.2.4 ResourceQuery

Type: Interface
Package: ResourceDataSink
This is an interface through which a client can define Queries on Resources so as to filter the information returned. Classes implementing the interface provide means to set the query parameters (such as a constructor).

Table 10.5 – Methods of Interface ResourceQuery

Method	Notes	Parameters
satisfies()	This operation is the client's implementation of a filtering query for Resources	Resource **resource** The resource to which to apply the filter

10.4.2.5 ResourceSinkListener

Type: Interface
Package: ResourceDataSink

This is an interface for clients to implement callback to receive information on changes to Resources.

Table 10.6 – Methods of Interface ResourceSinkListener

Method	Notes	Parameters
dataChanged()	This operation is implemented by the client to process the dataChanged callback. Multiple changes can be notified through a single invocation.	ResourceChangedEventList **eventList** The list of resource changes recevied by the listener

10.5 PlanExecutionRecommendations

Parent Package: ServiceModel

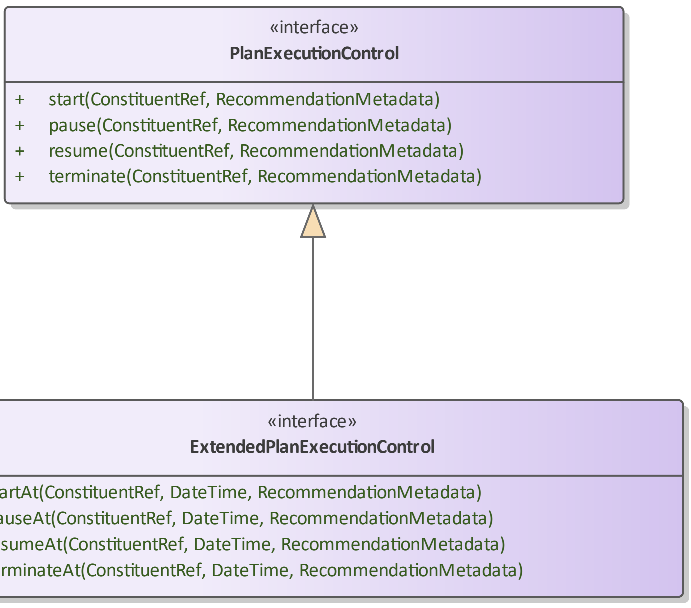

«interface»
PlanExecutionControl
+ start(ConstituentRef, RecommendationMetadata)
+ pause(ConstituentRef, RecommendationMetadata)
+ resume(ConstituentRef, RecommendationMetadata)
+ terminate(ConstituentRef, RecommendationMetadata)
«interface»
ExtendedPlanExecutionControl
+ startAt(ConstituentRef, DateTime, RecommendationMetadata)
+ pauseAt(ConstituentRef, DateTime, RecommendationMetadata)
+ resumeAt(ConstituentRef, DateTime, RecommendationMetadata)
+ terminateAt(ConstituentRef, DateTime, RecommendationMetadata)

Figure 10.18 ActionControlRecommendation (Class diagram)

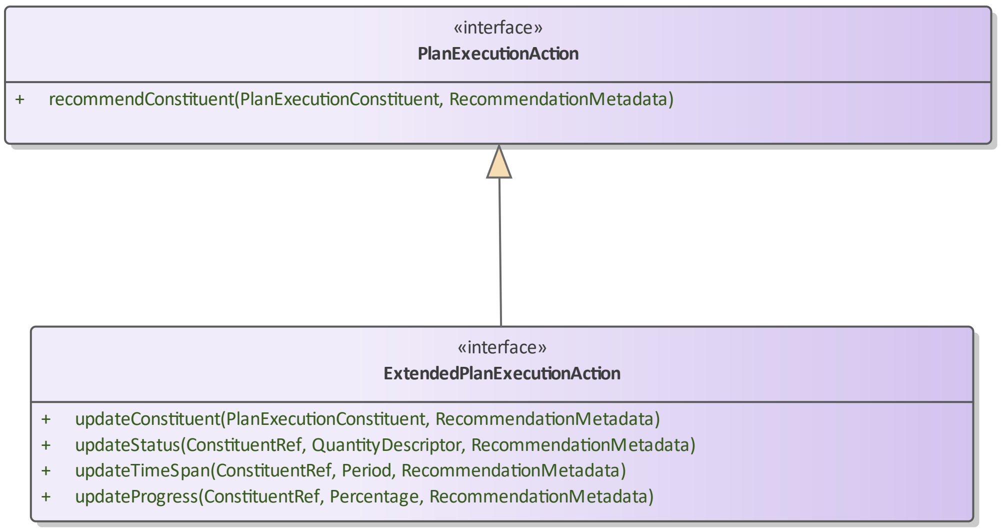

«interface»
PlanExecutionAction
+ recommendConstituent(PlanExecutionConstituent, RecommendationMetadata)
«interface»
ExtendedPlanExecutionAction
+ updateConstituent(PlanExecutionConstituent, RecommendationMetadata)
+ updateStatus(ConstituentRef, QuantityDescriptor, RecommendationMetadata)
+ updateTimeSpan(ConstituentRef, Period, RecommendationMetadata)
+ updateProgress(ConstituentRef, Percentage, RecommendationMetadata)

Figure 10.19 ActionRecommendation (Class diagram)

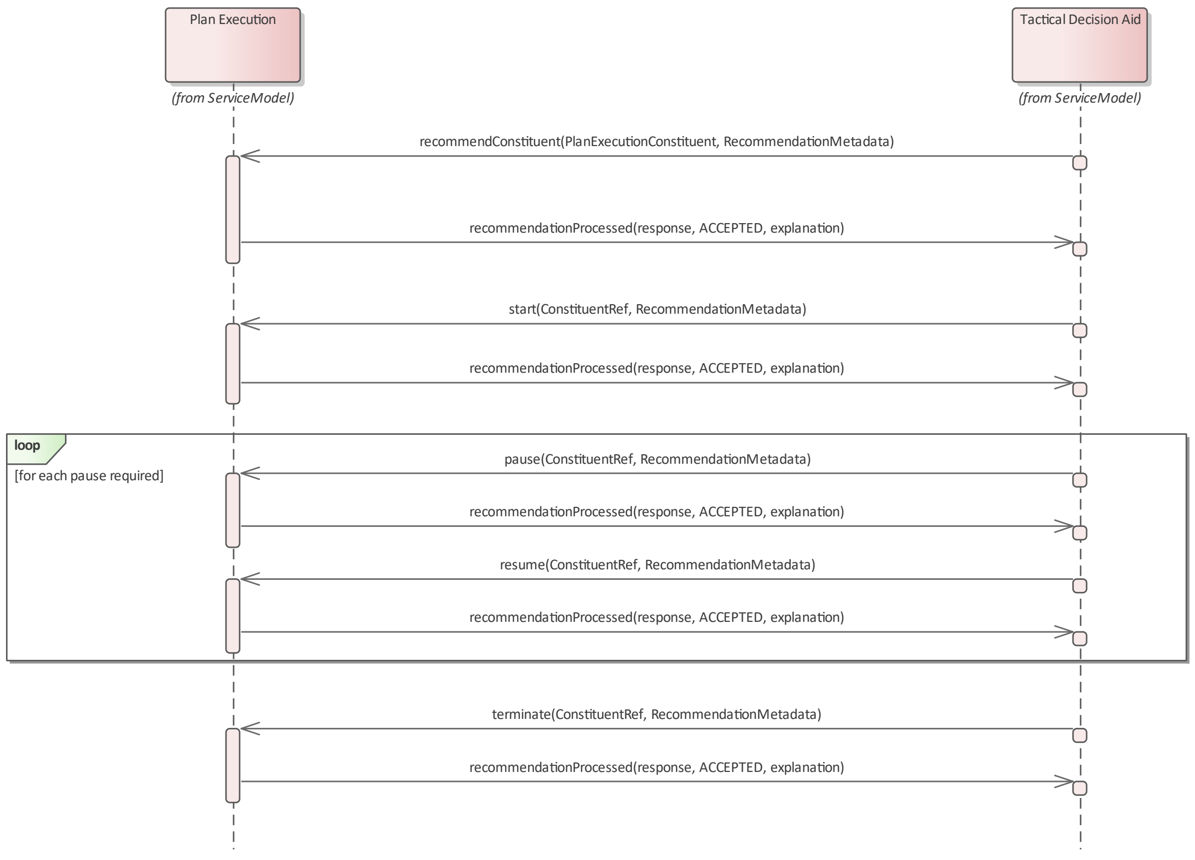

Plan Execution
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
recommendConstituent(PlanExecutionConstituent, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
start(ConstituentRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
loop
[for each pause required]
pause(ConstituentRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
resume(ConstituentRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
terminate(ConstituentRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)

Figure 10.20 PlanExecutionRecommendations (Interaction diagram)

Use of the PlanExecutionAction and PlanExecutionControl interfaces to recommend and then control the execution of a plan constituent. The ConstituentRef references the PlanExecutionConstituent.

In this scenario all recommendations are successfully accepted. It is valid for a Tactical Decision Aid to make additional recommendations before a recommendationProcessed response is received.

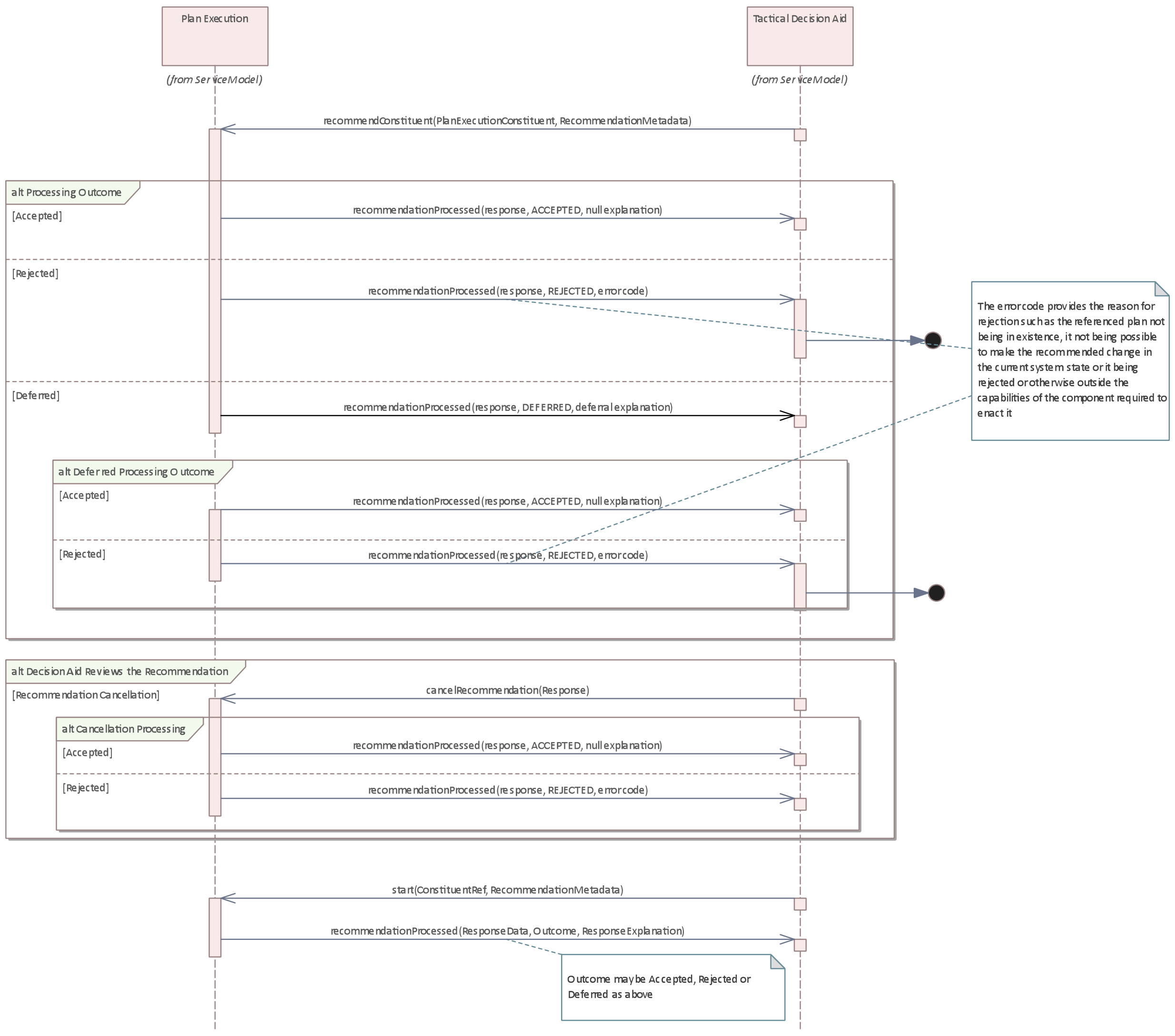

Plan Execution
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
recommendConstituent(PlanExecutionConstituent, RecommendationMetadata)
alt Processing Outcome
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)
The errorcode provides the reason for rejection such as the referenced plan not being in existence, it not being possible to make the recommended change in the current system state or it being rejected or otherwise outside the capabilities of the component required to enact it
[Deferred]
recommendationProcessed(response, DEFERRED, deferral explanation)
alt Deferred Processing Outcome
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)
alt Decision Aid Reviews the Recommendation
[Recommendation Cancellation]
cancelRecommendation(Response)
alt Cancellation Processing
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)
start(ConstituentRef, RecommendationMetadata)
recommendationProcessed(ResponseData, Outcome, ResponseExplanation)
Outcome may be Accepted, Rejected or Deferred as above

Figure 10.21 PlanExecutionRecommendations – Alternate Flows (Interaction diagram)

Use of the PlanExecutionAction and PlanExecutionControl interfaces to recommend and then control the execution of a plan constituent. In this set of scenarios not all recommendations are successfully accepted. Decision Aids may request cancellation of recommendations on subsequent review of evolving information; whether such requests are capable of being successfully processed is implementation and scenario dependent.

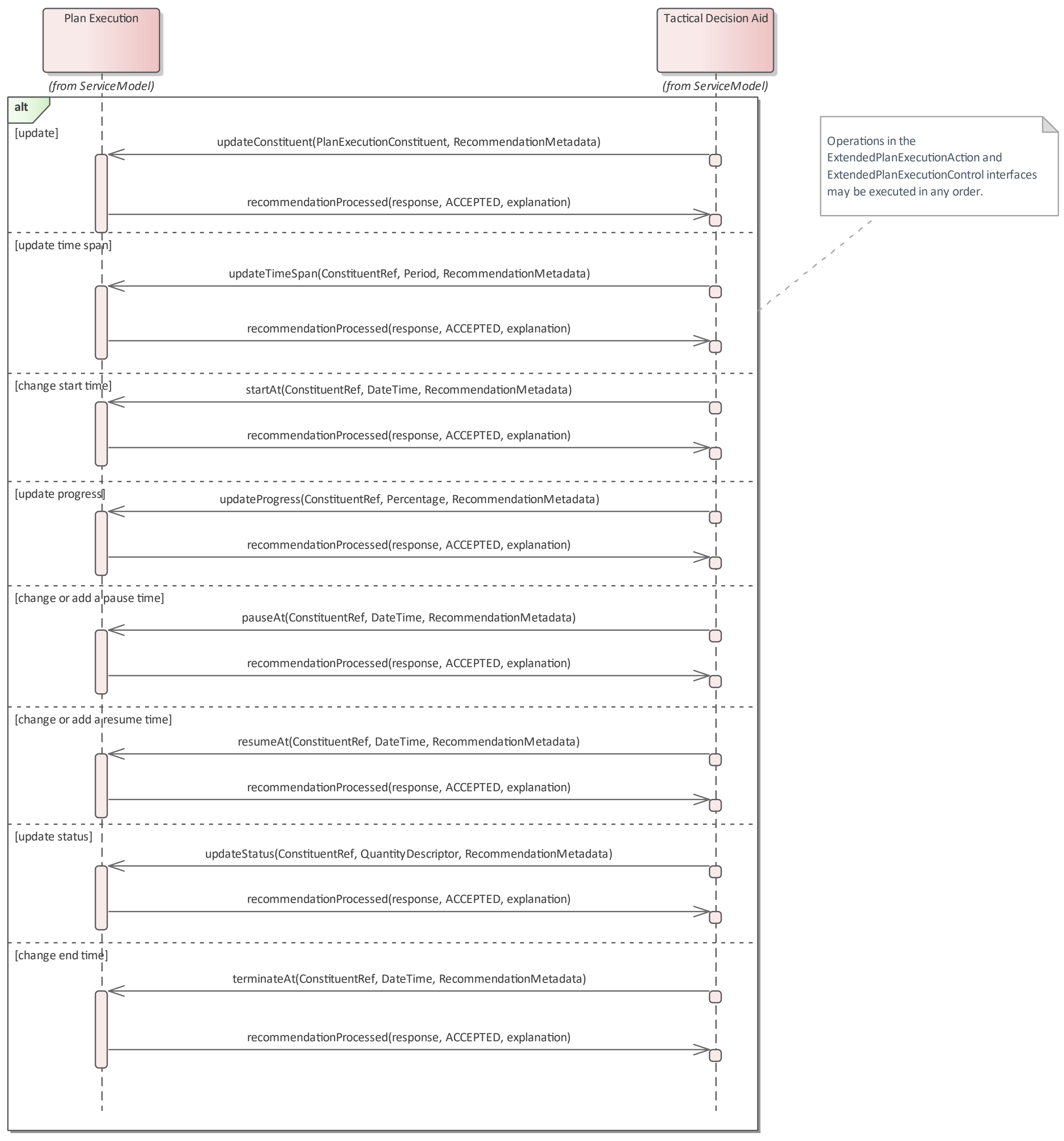

Plan Execution
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
alt
[update]
updateConstituent(PlanExecutionConstituent, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
[update time span]
updateTimeSpan(ConstituentRef, Period, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
[change start time]
startAt(ConstituentRef, DateTime, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
[update progress]
updateProgress(ConstituentRef, Percentage, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
[change or add a pause time]
pauseAt(ConstituentRef, DateTime, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
[change or add a resume time]
resumeAt(ConstituentRef, DateTime, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
[update status]
updateStatus(ConstituentRef, QuantityDescriptor, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
[change end time]
terminateAt(ConstituentRef, DateTime, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
Operations in the ExtendedPlanExecutionAction and ExtendedPlanExecutionControl interfaces may be executed in any order.

Figure 10.22 ExtendedPlanExecutionRecommendations (Interaction diagram)

Use of the ExtendedPlanExecutionAction and ExtendedPlanExecutionControl interfaces to recommend update of the content, timing, status and progress or a plan constituent as well as to control the its future execution. It is valid for the recommendations in this scenario to be made in any order, omitted or superceded with subsequent recommendations. In this scenario all recommendations are successfully accepted. It is valid for a Tactical Decision Aid to make additional recommendations before a recommendationProcessed response is received.

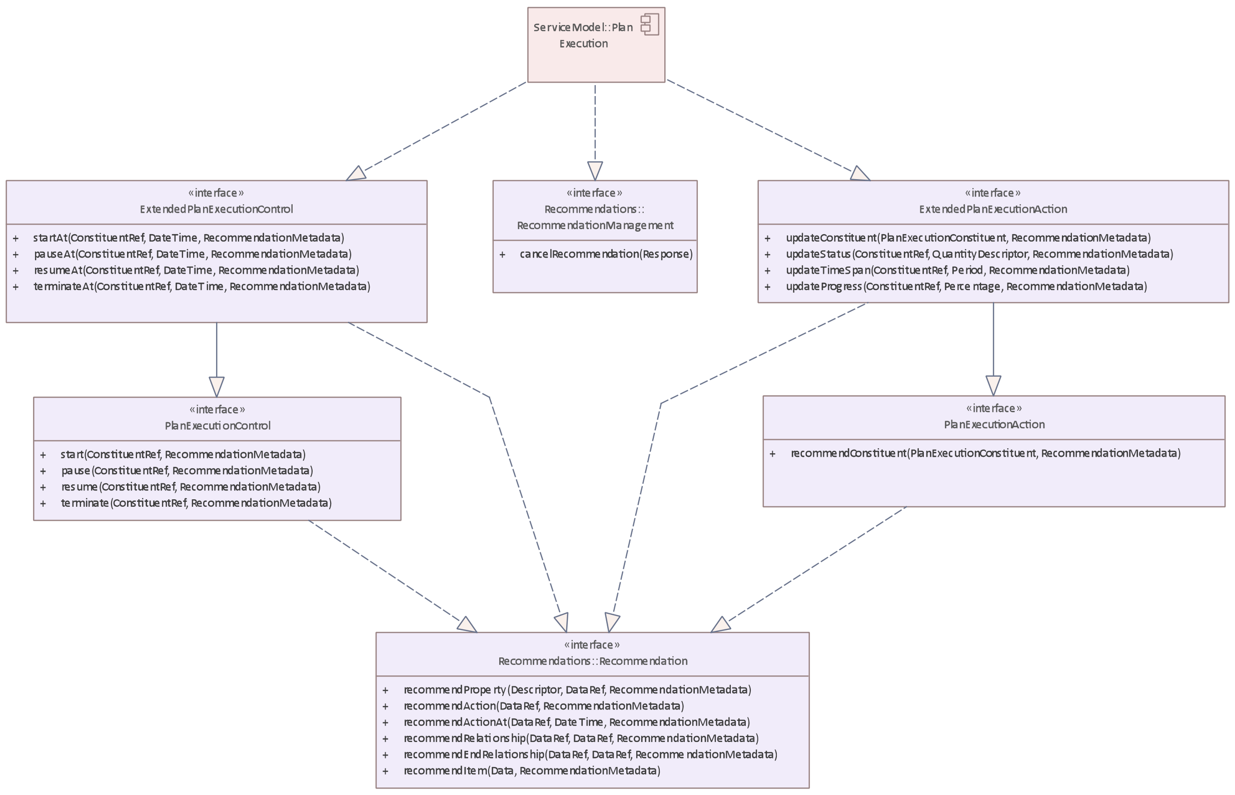

ServiceModel::Plan Execution
«interface» ExtendedPlanExecutionControl
+ startAt(ConstituentRef, DateTime, RecommendationMetadata)
+ pauseAt(ConstituentRef, DateTime, RecommendationMetadata)
+ resumeAt(ConstituentRef, DateTime, RecommendationMetadata)
+ terminateAt(ConstituentRef, DateTime, RecommendationMetadata)
«interface» Recommendations::RecommendationManagement
+ cancelRecommendation(Response)
«interface» ExtendedPlanExecutionAction
+ updateConstituent(PlanExecutionConstituent, RecommendationMetadata)
+ updateStatus(ConstituentRef, QuantityDescriptor, RecommendationMetadata)
+ updateTimeSpan(ConstituentRef, Period, RecommendationMetadata)
+ updateProgress(ConstituentRef, Percentage, RecommendationMetadata)
«interface» PlanExecutionControl
+ start(ConstituentRef, RecommendationMetadata)
+ pause(ConstituentRef, RecommendationMetadata)
+ resume(ConstituentRef, RecommendationMetadata)
+ terminate(ConstituentRef, RecommendationMetadata)
«interface» PlanExecutionAction
+ recommendConstituent(PlanExecutionConstituent, RecommendationMetadata)
«interface» Recommendations::Recommendation
+ recommendProperty(Descriptor, DataRef, RecommendationMetadata)
+ recommendAction(DataRef, RecommendationMetadata)
+ recommendActionAt(DataRef, DateTime, RecommendationMetadata)
+ recommendRelationship(DataRef, DataRef, RecommendationMetadata)
+ recommendEndRelationship(DataRef, DataRef, RecommendationMetadata)
+ recommendItem(Data, RecommendationMetadata)

Figure 10.23 PlanExecutionRecommendations Service Mapping (Component diagram)

10.5.1 PlanExecutionAction

Type: Interface
Package: PlanExecutionRecommendations
This interface allows client tactical decision aids to make recommendations to enact tactical Plans.
Referenced instances must exist. Therefore decision aids should first create any referenced entities, then recommend plan(s), then any sub-plans, then contributing task objectives, then implementing resource tasking recommendations.
All Recommendation operations on the PlanExecutionAction interface receive a PlanExecutionResponse instance in the callback.
It is invalid to recommend a constituent that already exists. That is, a PlanExecutionConstituent is returned for the ConstituentRef through the PlanDataSink interface.

ReferencedClass = Plan

Table 10.7 – Methods of Interface PlanExecutionAction

Method	Notes	Parameters
recommendConstituent()	This is the operation to invoke to recommend a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking.	PlanExecutionConstituent **plan** The constituent of plan execution being recommended RecommendationMetadata **recommendation** Qualifying information relating to the recommendation

10.5.1.1 MessageEnd

Type: MessageEnd
Package: PlanExecutionRecommendations

10.5.1.2 MessageEnd

Type: MessageEnd
Package: PlanExecutionRecommendations

10.5.1.3 MessageEnd

Type: MessageEnd
Package: PlanExecutionRecommendations

10.5.1.4 MessageEnd

Type: MessageEnd
Package: PlanExecutionRecommendations

10.5.2 PlanExecutionControl

Type: Interface
Package: PlanExecutionRecommendations
This interface allows client tactical decision aids to make recommendations to control the execution of tactical plan-constituents. All Recommendation operations on the PlanExecutionControl interface receive a PlanExecutionResponse instance in the callback.
It is invalid to recommend a change to the execution of a constituent that does not exist. That is, no PlanExecutionConstituent is returned for the ConstituentRef through the PlanDataSink interface.

Table 10.8 – Methods of Interface PlanExecutionControl

Method	Notes	Parameters
start()	This is the operation to invoke to recommend that a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking is started immediately. The constituent must	ConstituentRef **id** A reference to the constituent of plan execution for which the action is being recommended RecommendationMetadata **recommendation** Qualifying

	not have previously been started.	information relating to the recommendation
pause()	This is the operation to invoke to recommend that a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking is paused immediately.	ConstituentRef **id** The list of plan changes recevied by the listener RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
resume()	This is the operation to invoke to recommend that a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking is resumed immediately. The constituent must have previously been paused.	ConstituentRef **id** The list of plan changes recevied by the listener RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
terminate()	This is the operation to invoke to recommend that a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking is terminated immediately. The constituent must have previously been started.	ConstituentRef **id** The list of plan changes recevied by the listener RecommendationMetadata **recommendation** Qualifying information relating to the recommendation

10.5.3 ExtendedPlanExecutionAction

Type: Interface
Package: PlanExecutionRecommendations
This interface allows client tactical decision aids to make recommendations to update tactical Plans in whole or part.
All Recommendation operations on the ExtendedPlanAction interface receive a PlanResponse instance in the callback.
It is invalid to recommend an update to constituent that does not exist. That is, no PlanExecutionConstituent is returned for the ConstituentRef through the PlanDataSink interface.

Table 10.9 – Methods of Interface ExtendedPlanExecutionAction

Method	Notes	Parameters
updateConstituent()	This is the operation to invoke to recommend the update of a whole Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking.	PlanExecutionConstituent **planExecutionConstituent** The new values recommended for the plan execution constituent. RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
updateStatus()	This is the operation to invoke to recommend a change of a status to a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking. It is invalid to recommend an update to constituent that does not exist. That is, there is no instance within	ConstituentRef **id** A reference to the plan constituent QuantityDescriptor **status** The status value to update to RecommendationMetadata **recommendation** Qualifying information relating to the recommendation

	the Plan Execution component with the specified id. .	
updateTimeSpan()	This is the operation to invoke to recommend a change of a time span for a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking. It is invalid to recommend an update to constituent that does not exist. That is, there is no instance within the Plan Execution component with the specified id. .	ConstituentRef **id** A reference to the plan constituent Period **timeSpan** The time span value to update to RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
updateProgress()	This is the operation to invoke to recommend an update to the progress achieved for a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking. It is invalid to recommend an update to constituent that does not exist. That is, there is no instance within the Plan Execution component with the specified id.	ConstituentRef **id** A reference to the plan constituent Percentage **progress** The progress value to update to RecommendationMetadata **recommendation** Metadata pertaining to the recommendation

10.5.4 ExtendedPlanExecutionControl

Type: Interface
Package: PlanExecutionRecommendations
This interface allows client tactical decision aids to make recommendations to control the future execution of tactical plan-constituents. All Recommendation operations on the ExtendedPlanExecutionControl interface receive a PlanExecutionResponse instance in the callback.
It is invalid to recommend a change to the execution of a constituent that does not exist. That is, no PlanExecutionConstituent is returned for the ConstituentRef through the PlanDataSink interface.

Table 10.10 – Methods of Interface ExtendedPlanExecutionControl

Method	Notes	Parameters
startAt()	This is the operation to invoke to recommend that a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking is started at a future time. This must be before the end of it's time-span	ConstituentRef **id** A reference to the constituent of plan execution for which the action is being recommended DateTime **time** The time at which it is recommended to start executing the planning constituent RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
pauseAt()	This is the operation to invoke to recommend that a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking is paused at a	ConstituentRef **id** A reference to the constituent of plan execution for which the action is being recommended DateTime **time** The time at which it

	future time. This must be within it's time-span.	is recommended to pause execution of the planning constituent RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
resumeAt()	This is the operation to invoke to recommend that a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking is resumed at a future time. This must be within it's time-span.	ConstituentRef **id** A reference to the constituent of plan execution for which the action is being recommended DateTime **time** The time at which it is recommended to resume execution of the planning constituent RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
terminateAt()	This is the operation to invoke to recommend that a Plan Execution Constituent specialization such as a Plan, TaskObjective or ResourceTasking is terminated at a future time. This must be after the start of it's time-span.	ConstituentRef **id** A reference to the constituent of plan execution for which the action is being recommended DateTime **time** The time at which it is recommended to terminate the execution of the planning constituent RecommendationMetadata **recommendation** Qualifying information relating to the recommendation

10.6 TacticalPictureInformation

Parent Package: ServiceModel

Interfaces to allow Tactical Decision Aids to receive tactical picture information are contained in the TACSIT Data Exchange specification.
TacticalPictureInformation requirements are satisfied by TACSIT Data Exchange services.

10.7 TacticalPictureRecommendations

Parent Package: ServiceModel

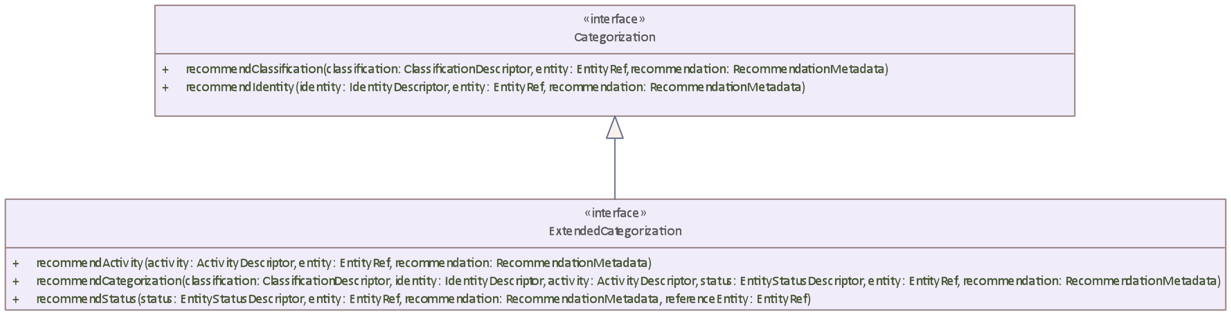

«interface»
Categorization
+ recommendClassification(classification: ClassificationDescriptor, entity: EntityRef, recommendation: RecommendationMetadata)
+ recommendIdentity(identity: IdentityDescriptor, entity: EntityRef, recommendation: RecommendationMetadata)
«interface»
ExtendedCategorization
+ recommendActivity(activity: ActivityDescriptor, entity: EntityRef, recommendation: RecommendationMetadata)
+ recommendCategorization(classification: ClassificationDescriptor, identity: IdentityDescriptor, activity: ActivityDescriptor, status: EntityStatusDescriptor, entity: EntityRef, recommendation: RecommendationMetadata)
+ recommendStatus(status: EntityStatusDescriptor, entity: EntityRef, recommendation: RecommendationMetadata, referenceEntity: EntityRef)

Figure 10.24 CategorizationRecommendation (Class diagram)

This is the interface for making Categorization Recommendations for Entities in the tactical picture.

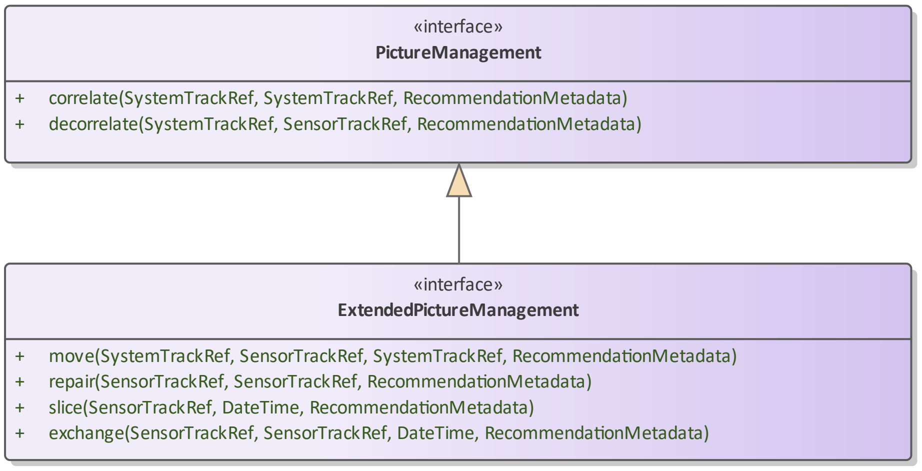

«interface»
PictureManagement
+ correlate(SystemTrackRef, SystemTrackRef, RecommendationMetadata)
+ decorrelate(SystemTrackRef, SensorTrackRef, RecommendationMetadata)
«interface»
ExtendedPictureManagement
+ move(SystemTrackRef, SensorTrackRef, SystemTrackRef, RecommendationMetadata)
+ repair(SensorTrackRef, SensorTrackRef, RecommendationMetadata)
+ slice(SensorTrackRef, DateTime, RecommendationMetadata)
+ exchange(SensorTrackRef, SensorTrackRef, DateTime, RecommendationMetadata)

Figure 10.25 PictureManagementRecommendation (Class diagram)

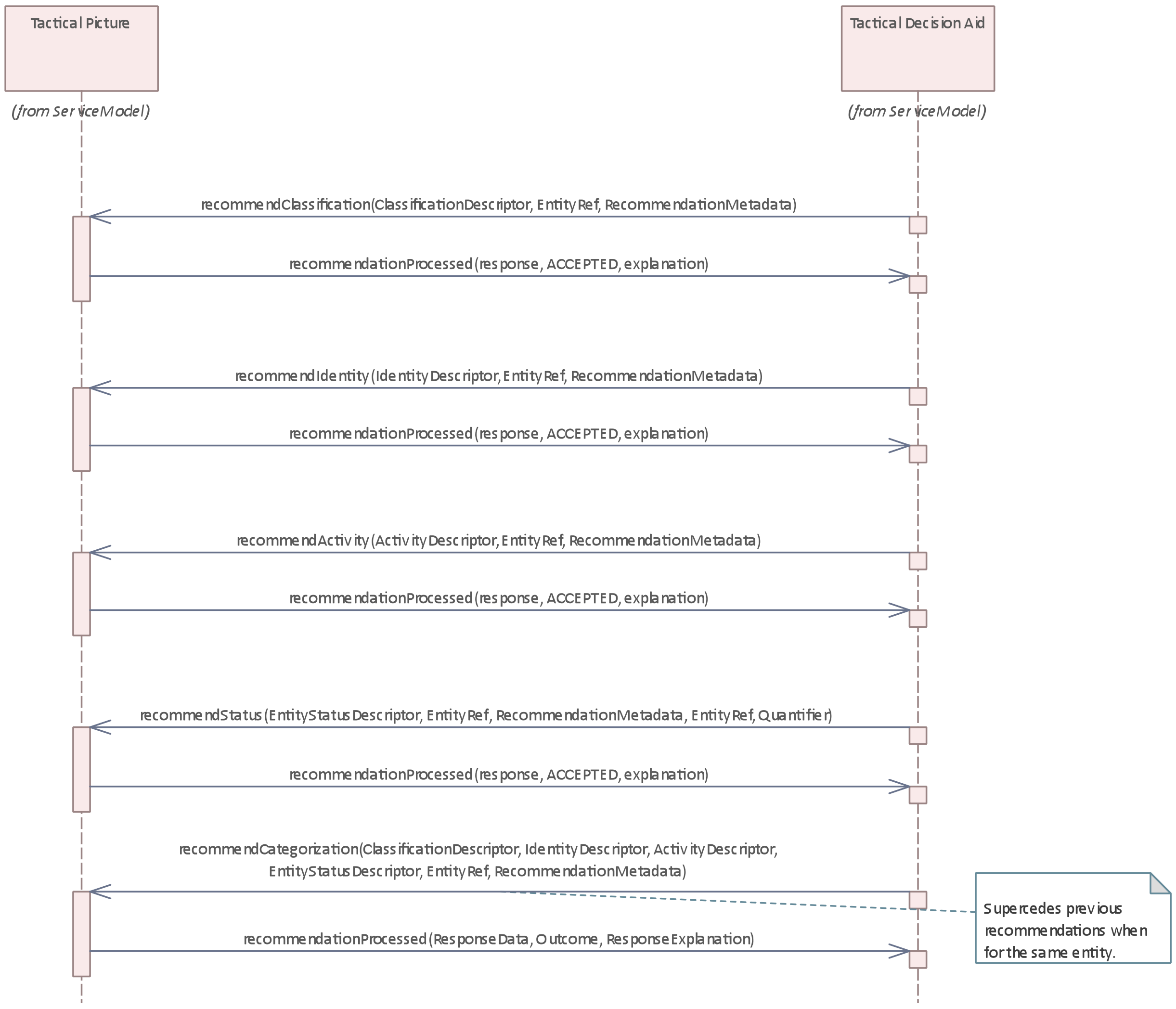

Tactical Picture
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
recommendClassification(ClassificationDescriptor, EntityRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
recommendIdentity(IdentityDescriptor, EntityRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
recommendActivity(ActivityDescriptor, EntityRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
recommendStatus(EntityStatusDescriptor, EntityRef, RecommendationMetadata, EntityRef, Quantifier)
recommendationProcessed(response, ACCEPTED, explanation)
recommendCategorization(ClassificationDescriptor, IdentityDescriptor, ActivityDescriptor, EntityStatusDescriptor, EntityRef, RecommendationMetadata)
recommendationProcessed(ResponseData, Outcome, ResponseExplanation)
Supercedes previous recommendations when for the same entity.

Figure 10.26 TacticalPictureRecommendations – Categorization (Interaction diagram)

Use of the Categorization and ExtendedCategorization interfaces to recommend categories for entities. In this scenario all recommendations are successfully accepted. It is valid for a Tactical Decision Aid to make additional recommendations before a recommendationProcessed response is received.

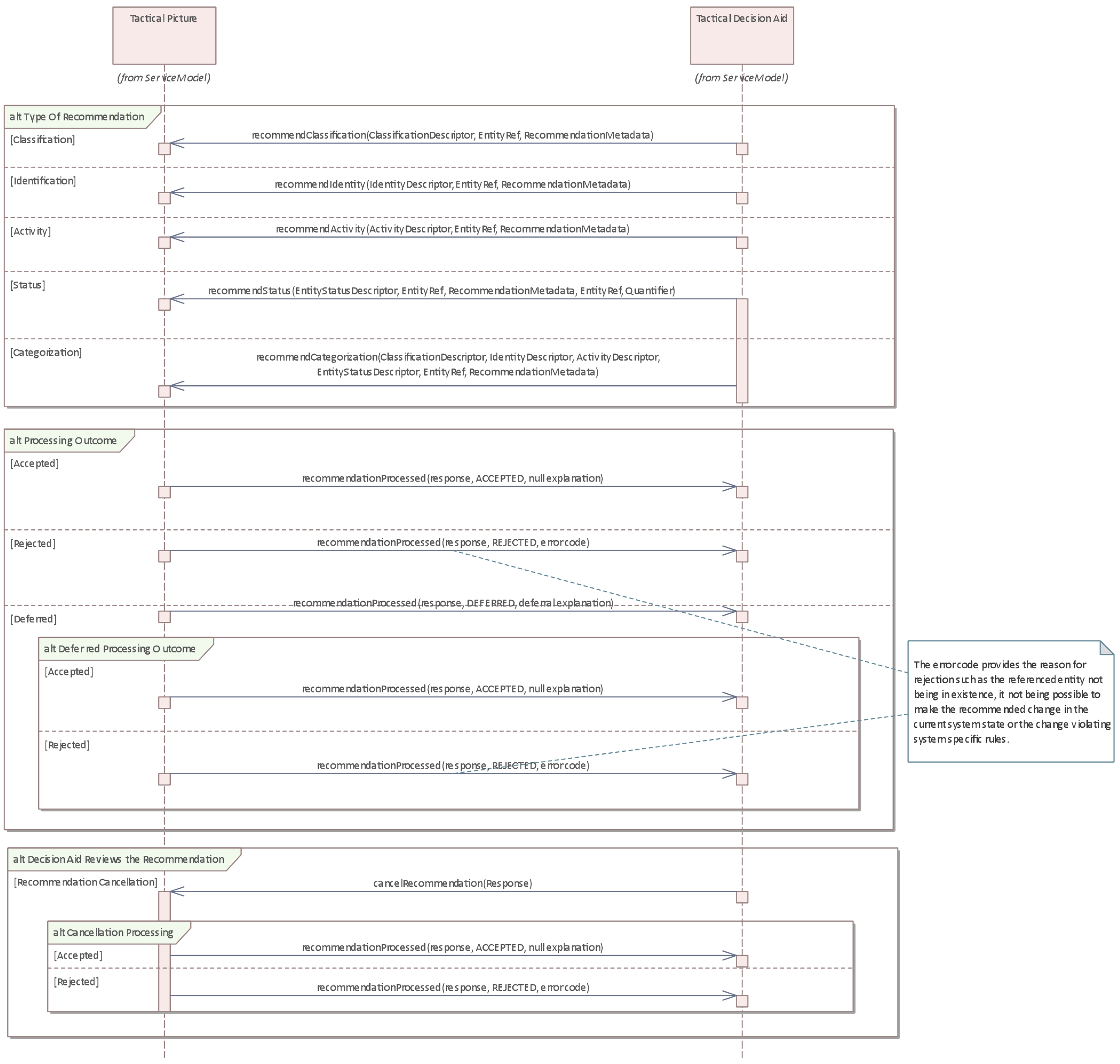

Tactical Picture
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
alt Type Of Recommendation
[Classification]
recommendClassification(ClassificationDescriptor, EntityRef, RecommendationMetadata)
[Identification]
recommendIdentity(IdentityDescriptor, EntityRef, RecommendationMetadata)
[Activity]
recommendActivity(ActivityDescriptor, EntityRef, RecommendationMetadata)
[Status]
recommendStatus(EntityStatusDescriptor, EntityRef, RecommendationMetadata, EntityRef, Quantifier)
[Categorization]
recommendCategorization(ClassificationDescriptor, IdentityDescriptor, ActivityDescriptor, EntityStatusDescriptor, EntityRef, RecommendationMetadata)
alt Processing Outcome
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)
[Deferred]
recommendationProcessed(response, DEFERRED, deferral explanation)
alt Deferred Processing Outcome
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)
The errorcode provides the reason for rejection such as the referenced entity not being in existence, it not being possible to make the recommended change in the current system state or the change violating system specific rules.
alt Decision Aid Reviews the Recommendation
[Recommendation Cancellation]
cancelRecommendation(Response)
alt Cancellation Processing
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)

Figure 10.27 TacticalPictureRecommendations – Categorization – Alternate Flow (Interaction diagram)

Use of the Categorization and ExtendedCategorization interfaces to recommend categories for entities. In this scenario not all recommendations are successfully accepted. Alternate outcome processing is equivalent for all operations on the Categorization and ExtendedCategorization interfaces. Decision Aids may request cancellation of recommendations on subsequent review of evolving information; whether such requests are capable of being successfully processed is implementation and scenario dependent.

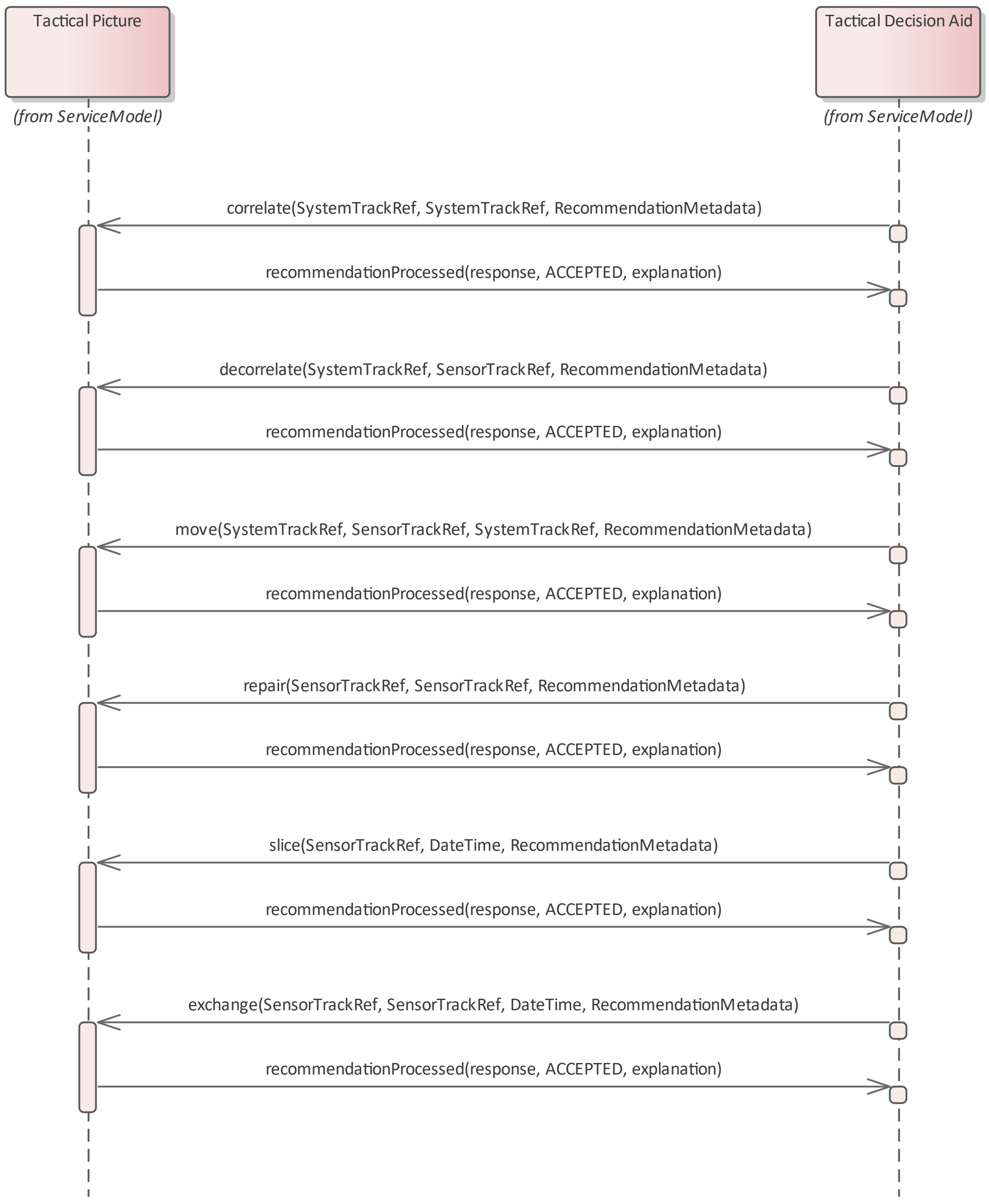

Tactical Picture
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
correlate(SystemTrackRef, SystemTrackRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
decorrelate(SystemTrackRef, SensorTrackRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
move(SystemTrackRef, SensorTrackRef, SystemTrackRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
repair(SensorTrackRef, SensorTrackRef, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
slice(SensorTrackRef, DateTime, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)
exchange(SensorTrackRef, SensorTrackRef, DateTime, RecommendationMetadata)
recommendationProcessed(response, ACCEPTED, explanation)

Figure 10.28 TacticalPictureRecommendations – PictureManagement (Interaction diagram)

Use of the PictureManagement and ExtendedPictureManagement interfaces to recommend changes to the relationships between entities. In this scenario all recommendations are successfully accepted. It is valid for a Tactical Decision Aid to make additional recommendations before a recommendationProcessed response is received.

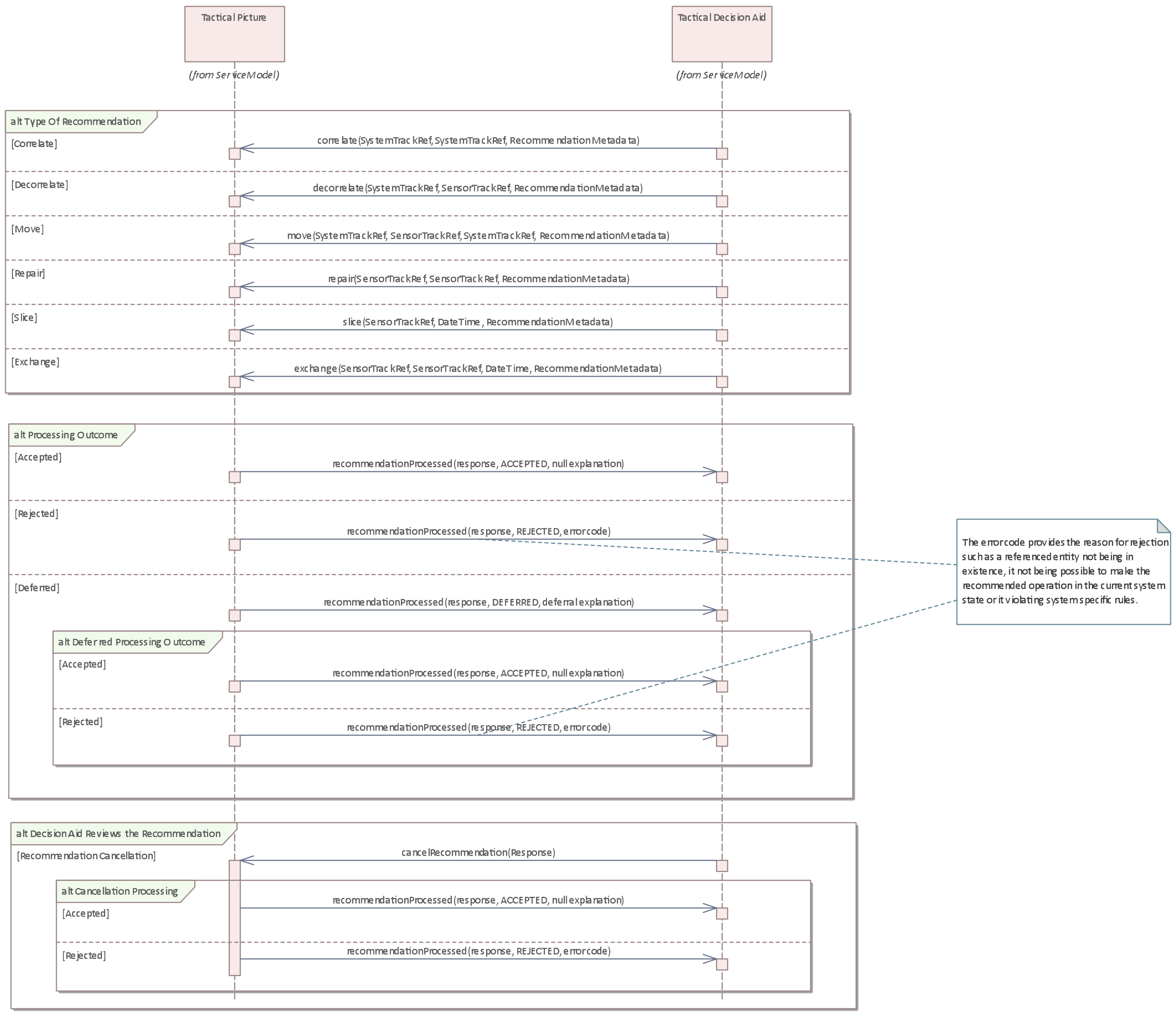

Tactical Picture
(from ServiceModel)
Tactical Decision Aid
(from ServiceModel)
alt Type Of Recommendation
[Correlate]
correlate(SystemTrackRef, SystemTrackRef, RecommendationMetadata)
[Decorrelate]
decorrelate(SystemTrackRef, SensorTrackRef, RecommendationMetadata)
[Move]
move(SystemTrackRef, SensorTrackRef, SystemTrackRef, RecommendationMetadata)
[Repair]
repair(SensorTrackRef, SensorTrackRef, RecommendationMetadata)
[Slice]
slice(SensorTrackRef, DateTime, RecommendationMetadata)
[Exchange]
exchange(SensorTrackRef, SensorTrackRef, DateTime, RecommendationMetadata)
alt Processing Outcome
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)
[Deferred]
recommendationProcessed(response, DEFERRED, deferral explanation)
alt Deferred Processing Outcome
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)
The errorcode provides the reason for rejection such as a referenced entity not being in existence, it not being possible to make the recommended operation in the current system state or it violating system specific rules.
alt Decision Aid Reviews the Recommendation
[Recommendation Cancellation]
cancelRecommendation(Response)
alt Cancellation Processing
[Accepted]
recommendationProcessed(response, ACCEPTED, null explanation)
[Rejected]
recommendationProcessed(response, REJECTED, errorcode)

Figure 10.29 TacticalPictureRecommendations – PictureManagement – Alternate Flow (Interaction diagram)

Use of the PictureManagement and ExtendedPictureManagement interfaces to recommend categories for entities. In this scenario not all recommendations are successfully accepted. Alternate outcome processing is equivalent for all operations on the PictureManagement and ExtendedPictureManagement interfaces. Decision Aids may request cancellation of recommendations on subsequent review of evolving information; whether such requests are capable of being successfully processed is implementation and scenario dependent.

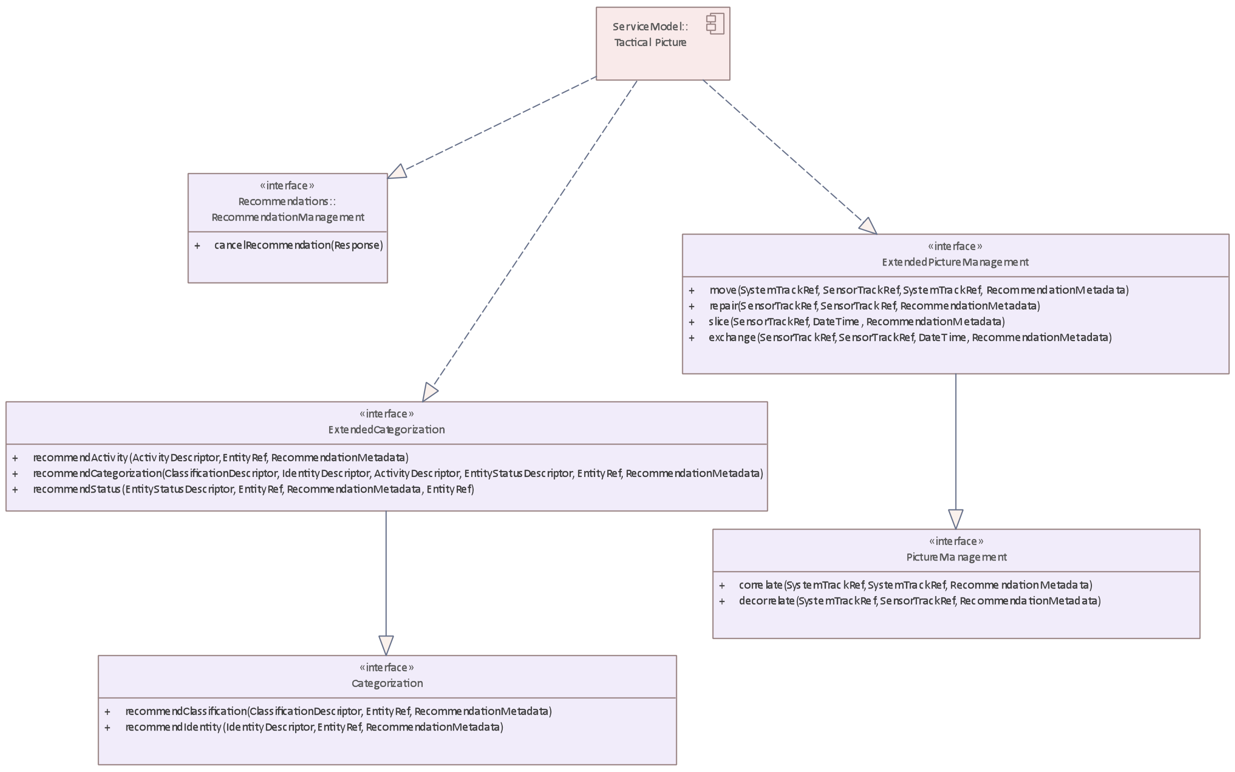

ServiceModel::
Tactical Picture
«interface»
Recommendations::
RecommendationManagement
+ cancelRecommendation(Response)
«interface»
ExtendedPictureManagement
+ move(SystemTrackRef, SensorTrackRef, SystemTrackRef, RecommendationMetadata)
+ repair(SensorTrackRef, SensorTrackRef, RecommendationMetadata)
+ slice(SensorTrackRef, DateTime , RecommendationMetadata)
+ exchange(SensorTrackRef, SensorTrackRef, DateTime, RecommendationMetadata)
«interface»
ExtendedCategorization
+ recommendActivity(ActivityDescriptor, EntityRef, RecommendationMetadata)
+ recommendCategorization(ClassificationDescriptor, IdentityDescriptor, ActivityDescriptor, EntityStatusDescriptor, EntityRef, RecommendationMetadata)
+ recommendStatus(EntityStatusDescriptor, EntityRef, RecommendationMetadata, EntityRef)
«interface»
PictureManagement
+ correlate(SystemTrackRef, SystemTrackRef, RecommendationMetadata)
+ decorrelate(SystemTrackRef, SensorTrackRef, RecommendationMetadata)
«interface»
Categorization
+ recommendClassification(ClassificationDescriptor, EntityRef, RecommendationMetadata)
+ recommendIdentity(IdentityDescriptor, EntityRef, RecommendationMetadata)

Figure 10.30 TacticalPictureRecommendations Service Mapping (Component diagram)

10.7.1 Categorization

Type: Interface
Package: TacticalPictureRecommendations
This interface allows client tactical decision aids to make recommendations to categorize Entities in the tactical picture. I.e. recommendation relating to Entity Categorization data as defined by the TACSIT Data Exchange (TEX) standard. This interface supports recommendations relating to the most common tactical categorization decisions and hence those recommendations most likely to be generated by decision aids.
All Recommendation operations on the Categorization interface receive a CategorizationResponse instance in the callback.
It is invalid to recommend a categorization for an Entity that does not exist. That is, no Entity is returned for the EntityRef through the TEX DataSink interface.

Table 10.11 – Methods of Interface Categorization

Method	Notes	Parameters
recommendClassification()	This is an operation to invoke to make a Classification Recommendation. Classification refers to the kind of platform or vehicle that the Entity represents. Examples include truck, ferry, submarine, helicopter and satellite.	ClassificationDescriptor **classification** The classification being recommended EntityRef **entity** The entity to which the classification applies RecommendationMetadata **recommendation** Qualifying information relating to the

		recommendation
recommendIdentity()	This is an operation to invoke to make an Identification Recommendation. Identification refers to the allegiance or ownership of platform or vehicle that the Entity represents. This can be expressed, for example, as a hostility category (also known as standard identity) a nationality, an organization or personal identifier.	IdentityDescriptor **identity** The identity being recommended EntityRef **entity** The entity to which the identity applies RecommendationMetadata **recommendation** Qualifying information relating to the recommendation

10.7.2 ExtendedCategorization

Type: Interface
Package: TacticalPictureRecommendations
This interface allows client tactical decision aids to make specialized recommendations to categorize Entities in the tactical picture. I.e. recommendation relating to Entity Categorization data as defined by the TACSIT Data Exchange (TEX) standard. This interface supports recommendations relating to more advanced or specialized tactical categorization decisions and hence those recommendations that may not be generated by all decision aids.
All Recommendation operations on the ExtendedCategorization interface receive a CategorizationResponse instance in the callback.
It is invalid to recommend a categorization for an Entity that does not exist. That is, no Entity is returned for the EntityRef through the TEX DataSink interface.

Table 10.12 – Methods of Interface ExtendedCategorization

Method	Notes	Parameters
recommendActivity()	This is an operation to invoke to make an Activity Recommendation. Activity refers to the tasks currently being undertaken by the platform or vehicle that the Entity represents. Examples include Air Defence, Guard, Patrol, Reconnaissance, Refuel and Survey.	ActivityDescriptor **activity** The activity being recommended EntityRef **entity** The entity to which the activity applies RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
recommendCategorization()	This is an operation to invoke to make an atomic and coherent recommendation for multiple aspects of categorization (classification, identity, activity or status). A categorization recommendation for an entity supersedes preceding recommendations for individual aspects for that entity from the same Tactical Decision Aid.	ClassificationDescriptor **classification** [0..1] The classification aspect of the categorization being recommended IdentityDescriptor **identity** [0..1] The identity aspect of the categorization being recommended ActivityDescriptor **activity** [0..1] The activity aspect of the categorization being recommended EntityStatusDescriptor **status** [0..*] The set of additional status values contributing to the categorization EntityRef **entity** The entity to which the activity applies RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
recommendStatus()	This is an operation to invoke to make a Status Recommendation. Status refers to all aspects of the current tactical significance of the platform or vehicle that the Entity represents. Examples of status categories include: the extent to which the Entity poses a threat; the type of offensive action that the Entity is subject to, the outcome of offensive action and other emergencies.	EntityStatusDescriptor **status** The status being recommended RecommendationMetadata **recommendation** Qualifying information relating to the recommendation EntityRef **entity** The entity to which the status applies EntityRef **referenceEntity** [0..1] Optionally, the entity to which the status refers Quantifier **quantity** [0..1] Optionally, this parameter quantifies the status, for instance to determine relative priority of entities with the same status

10.7.3 PictureManagement

Type: Interface
Package: TacticalPictureRecommendations
This interface allows client tactical decision aids to make recommendations to manage the relationships between Entities in the tactical picture. I.e. recommendations relating to the constituent Entities of Groups as defined by the TACSIT Data Exchange (TEX) standard. This interface supports recommendations relating to

the most common tactical relation decisions and hence those recommendations most likely to be generated by decision aids.
Recommendation operations on the PictureManagement interface receive operation specific ResponseData specialization instances in the callback.
It is invalid to invoke an operation for an Entity that does not exist. That is, no Entity is returned for the EntityRef (SystemTrackRef or SensorTrackRef) through the TEX DataSink interface.

Table 10.13 – Methods of Interface PictureManagement

Method	Notes	Parameters
correlate()	This is the operation to invoke to make a Correlation Recommendation. Correlation refers to the determination that two or more sensor tracks correspond to the same object in the tactical environment. That object is to be represented by a single system track in an unambiguous tactical picture. Correlation relates the sensor tracks to the single system track. Note that it is typically possible for multiple sensors to observe and track the same object. Tactical Decision Aids receive a CorrelationResponse instance in the callback.	SystemTrackRef **receiver** The system track to be retained after the operation SystemTrackRef **donor** The system track to discard after the operation RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
decorrelate()	This is the operation to invoke to make a Decorrelation Recommendation. Decorrelation is the reverse of Correlation and is used to undo incorrect Correlations or to correct the case when a sensor has started to track a different object with the same sensor track. Tactical Decision Aids receive a DecorrelationResponse instance in the callback.	SystemTrackRef **systemTrack** SensorTrackRef **sensorTrack** RecommendationMetadata **recommendation** Qualifying information relating to the recommendation

10.7.4 ExtendedPictureManagement

Type: Interface
Package: TacticalPictureRecommendations

This interface allows client tactical decision aids to make specialized recommendations to manage the relationships between Entities in the tactical picture. I.e. recommendations relating to the constituent Entities of Groups as defined by the TACSIT Data Exchange (TEX) standard. This interface supports recommendations relating more advanced or specialized tactical relation decisions and hence those recommendations that may not be generated by all decision aids.
Recommendation operations on the ExtendedPictureManagement interface receive operation specific ResponseData specialization instances in the callback.
It is invalid to invoke an operation for an Entity that does not exist. That is, no Entity is returned for the EntityRef (SystemTrackRef or SensorTrackRef) through the TEX DataSink interface.

Table 10.14 – Methods of Interface ExtendedPictureManagement

Method	Notes	Parameters
move()	This is the operation to invoke to make a Move Recommendation. Move is a sequence of a Decorrelation followed by a Correlation and is used to correct the case when one sensor has started to track (with the same sensor track) a different object that is already being tracked by another sensor. The sensor track in question is Decorrelated from its original system track and Correlated with the system track that already exists for the new object that the sensor is actually tracking. Tactical Decision Aids receive a MoveResponse instance in the callback.	SystemTrackRef **receiver** The system track to which the sensor track is to be moved SensorTrackRef **sensorTrack** The sensor track to move to a different system track SystemTrackRef **donor** The system track the sensor track is to be moved from. To be a valid recommendation the donor track should be supported by tracks other than the sensor track being moved. RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
repair()	This is the operation to invoke to make a Repair Recommendation. Repair is an action on a sensor track's track report history that is used to make the track history continuous when a sensor has declared deletion of track before, later, starting to report the same real world object with a new sensor track. The original sensor track's history is added to the new sensor track's history. The implementation is such that the TEX (TACSIT Data Exchange) Entity History interface returns the complete history for the repaired track and no history for the deleted track as defined by the RepairResponse instance received in the callback.	SensorTrackRef **newSensorTrack** The track currently supported by the sensor SensorTrackRef **oldSensorTrack** The track previously supported by the sensor RecommendationMetadata **recommendation** Qualifying information relating to the recommendation
slice()	This is the operation to invoke to make a Slice Recommendation. Slice is an action on a sensor track's track report history that is used to make the track history discrete when a sensor has started to track a different real world object with the same sensor track. It is the inverse of Repair. The original part of sensor track's history is removed from sensor track's history and placed into a new sensor track. The implementation is such that the TEX (TACSIT Data Exchange) Entity	SensorTrackRef **sensorTrack** The sensor track whose history is to be sliced into new and old portions DateTime **sliceTime** The time at which to divide the tracks history RecommendationMetadata **recommendation** Qualifying information relating to the recommendation

	History interface returns the pre-slice history for the original track and the post-slice history for the new track as defined by the SliceResponse instance received in the callback.	
exchange()	This is the operation to invoke to make an Exchange Recommendation. Exchange is an action on a pair of sensor tracks' track report histories that is used to make the track histories coherent when a sensor has swapped the real-world objects that a pair of sensor tracks have been tracking. An exchange operation is a composition of two slice operations with a common slice-time followed by two repair operations. The sensor track parameters are semantically commutative: exchanging track-a with track-b is equivalent to exchanging track-b with track-a. Tactical Decision Aids receive an ExchangeResponse instance in the callback.	SensorTrackRef **sensorTrack1** The first sensor track to be exchanged SensorTrackRef **sensorTrack2** The second sensor track to be exchanged DateTime **exchangeTime** The time at which to exchange the tracks history RecommendationMetadata **recommendation** Qualifying information relating to the recommendation

This page intentionally left blank.

11 Domain Model Platform-Specific Models

11.1 DDS PSM

The DDS Data Model PSM defines a set of IDL files for the Data Model packages defined by the PIM. Topic types (i.e., IDL structs with keys) are defined for classes that classify a single parameter on an interface method. This avoids redundant indirection. Comments are added to the IDL files to reflect the mapping rules below.

The detailed rules for the MDA code generation from the Data Model 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. Generalization classes that have attributes are mapped to a struct containing a base struct for its common attributes and a variants union for the specialization attributes.
- The Duration datatype is mapped to an unsigned long long with the CORBA time representation (100s of nanoseconds since the start of the Gregorian Calendar).
- Other datatypes for real-valued quantities are mapped to a double.
- Navigable, by-reference, association roles are mapped to a datatype stereotyped as 'Reference', which has a 'refers to' relation with the destination class. Reference stereotyped datatypes are mapped to a string to represent an implementation specific unique id.
- Extensible Enumeration datatypes are mapped to a struct with a schemaPrefix string attribute and a value string attribute.

11.2 GraphQL PSM

The GraphQL PSM defines a single schema definition file for a combination of the Data Model and Service Model packages defined by the PIM. Classes from the Domain Model of the PIM are mapped to GraphQL types within the schema.

The detailed rules for the MDA code generation from the Data Model PIM to the GraphQL PSM schema are as follows:

- The PIM attributes are mapped to GraphQL attributes.
- PIM attributes with multiplicity 1 are mapped to non-nullable GraphQL attributes.
- Collections in the PIM are mapped to GraphQL arrays.
- By default, PIM classes are mapped to GraphQL object and input object types (input object types are required for services mapped to GraphQL mutations).
- Specialization / Generalization PIM relationships are mapped to GraphQL unions. Generalization classes that have attributes are mapped to a GraphQL type containing a base GraphQL object and input object type for its common attributes and a variants GraphQL union for the specialization attributes.
- The Duration datatype is mapped to a GraphQL Long datatype with the CORBA time representation (100s of nanoseconds since the start of the Gregorian Calendar).
- Other datatypes for real-valued quantities are mapped to a GraphQL Float.
- Navigable, by-reference, association roles are mapped to a datatype stereotyped as 'Reference', which has a 'refers to' relation with the destination class. Reference stereotyped datatypes are mapped to a string to represent an implementation specific unique id and a nullable (by default) attribute for the type of the destination class, so as to enable deep queries over a graph of instances.

- Extensible Enumeration datatypes are mapped to object and input object types with a schemaPrefix string attribute and a value string attribute.

12 Service Model Platform Specific Models

12.1 DDS PSM

The DDS Services PSM defines IDL files for each package defined in the Services PIM. For each method on each interface class an IDL struct for a DDS topic named for the method is generated; each parameter is mapped to an attribute of the IDL struct. This is unless there is only one attribute (of IDL struct stereotype) in which case the topic type is defined in the Domain Model (i.e., it corresponds to the single parameter's class). Return parameters, where specified, are also mapped to DDS Topics.

The PSM method for connecting to other components is through the creation of DDS Entities (specifically Participants, Data Readers and Data Writers).

Specific rules for the MDA code generation from the Service Model PIM to the DDS PSM IDL are as follows:

- The Response callback interface in the PIM is mapped to a struct with two keyed attributes of type short: clientId and requestId; The clientId identifies the Tactical Decision Aid making the request and the requestId distinguishes the recommendation from others made by the same Tactical Decision Aid.
- The DataSink pattern is mapped to a DDS topic type for the Data class. All interface methods are satisfied by built-in DDS API methods.
- From the Configuration interface, the getSupportMapping method is mapped to a topic for the input parameter and a topic for the return parameter and the isSupported method is mapped implicitly to DDS built-in discovery services.

12.2 GraphQL PSM

The GraphQL PSM defines a single schema definition file for a combination of the Data Model and Service Model packages defined by the PIM. The schema supports GraphQL clients for Tactical Decision Aids, Tactical Picture and Plan Execution components. Mutations are used to invoke PIM interface methods; queries and subscriptions are used to process those invocations.

The PSM method for connecting to other components is through the underlying HTTPS web service connection. Web-sockets are used for subscription callbacks.

Specific rules for the MDA code generation from the Service Model PIM to the GraphQL PSM schema are as follows:

- Each interface method in the Service Model is mapped to a (query) type, an input type and update type; these are for queries, mutations and subscriptions respectively.
- The GraphQL schema Query type support queries for any combination of interface methods in the Service Model.
- The GraphQL schema Mutation type supports invocation of single or multiple instances of any combination of interface methods in the Service Model.
- The GraphQL schema Subscription type supports subscription for any combination of interface methods in the Service Model.
- The Response callback interface in the PIM is mapped to a struct with two keyed attributes of type short: clientId and requestId; The clientId identifies the Tactical Decision AId making the request and the requestId distinguishes the recommendation from others made by the same Tactical Decision Aid.
- The DataSink pattern is mapped to the query, input and update types for the Data class. All interface methods are satisfied by built-in GraphQL features.
- From the Configuration interface, the getSupportMapping method is mapped to the query, input and update types for the input parameters and the query, input and update types for the return parameter and the isSupported method is mapped implicitly to GraphQL built-in discovery services.

This page intentionally left blank.

13 Platform Specific Models for Extensible Enumerations

The Tactical Decision Aids metamodel defines an Extensible Enumeration stereotype for a datatype that takes values from a finite set, where the set of values is not defined by the specification. Implementations define the valid set of values using platform specific mechanisms (see Data Model PSMs). This PSM defines normative alignment with other specifications by mapping Extensible Enumerations defined by this specification to definitions in other specifications.

Table 13.1 - Extensible Enumeration Mappings

Extensible Enumeration	Schema Prefix	Reference Specification	Reference Definition	Notes
Utils:: QuantityDescriptor	si	ISO 80000-1 :2009	N/A	SI units. Values are the unit symbols for base units, special symbols and derived symbols. E.g. "kg", "rad" and "m/s2"
TacticalPicture:: ActivityDescriptor	s5516.air	STANAG 5516 Ed 6	DFI 1798 DUI 001	Air activities. Values are the string representation of the DI bit code
TacticalPicture:: ActivityDescriptor	s5516.surf	STANAG 5516 Ed 6	DFI 1798 DUI 002	Surface activities. Values are the string representation of the DI bit code
TacticalPicture:: ActivityDescriptor	s5516.sub	STANAG 5516 Ed 6	DFI 1798 DUI 003	Subsurface activities. Values are the string representation of the DI bit code
TacticalPicture:: ActivityDescriptor	s5516.land	STANAG 5516 Ed 6	DFI 1798 DUI 004	Land activities. Values are the string representation of the DI bit code
TacticalPicture:: ActivityDescriptor	s5516.sp	STANAG 5516 Ed 6	DFI 1798 DUI 005	Space activities. Values are the string representation of the DI bit code
TacticalPicture:: ActivityDescriptor	s2525.atac	STANAG 2525 Rev D CN 1	TABLE A-XLII	Values are the string representation of the Code
TacticalPicture:: ActivityDescriptor	jc3iedm.atac	JC3IEDM v3.1.4	action-task-activity-code	Values are the capitalized abbreviations in the Physical Value column
TacticalPicture:: ActivityDescriptor	app6b.task	APP-6(B) June 2008	task graphics	Values are the specific one or two character code within the symbol id that is associated with the task type
TacticalPicture:: ActivityDescriptor	app6c.act	APP-6(C) May 2011	activity symbol table 6-3	Values (e.g. "Arrest") are from the function column of table 6-3
TacticalPicture:: ActivityDescriptor	app6c.task	APP-6(C) May 2011	mission tasks table 7-A-1	Values are the labels (e.g. "Ambush") from the control measure column of table 7-A-1
TacticalPicture:: ClassificationDescriptor	s5516.air.pl	STANAG 5516 Ed 6	DFI 1797 DUI 001	Air platforms. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516surf.pl	STANAG 5516 Ed 6	DFI 1797 DUI 002	Surface platforms. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516.sub.pl	STANAG 5516 Ed 6	DFI 1797 DUI 003	Subsurface platforms. Values are the string representation of the DI bit code

Extensible Enumeration	Schema Prefix	Reference Specification	Reference Definition	Notes
TacticalPicture:: ClassificationDescriptor	s5516.land.pl	STANAG 5516 Ed 6	DFI 1797 DUI 004	Land platforms. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516.sp.pl	STANAG 5516 Ed 6	DFI 1797 DUI 005	Space platforms. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516.air.st	STANAG 5516 Ed 6	DFI 804 DUI 001	Air specific type. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516.surf.st	STANAG 5516 Ed 6	DFI 808 DUI 001	Surface specific type. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516.sub.st	STANAG 5516 Ed 6	DFI 809 DUI 001	Subsurface specific type. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516.land.st	STANAG 5516 Ed 6	DFI 810 DUI 001	Land specific type. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516.sp.st	STANAG 5516 Ed 6	DFI 749 DUI 002	Space specific type. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	imo.id	IMO	N/A	The value to a vessel assigned by Lloyds Registry
TacticalPicture:: ClassificationDescriptor	imo.mmsi	IMO	N/A	The unique Maritime Mobile Service Identity (MMSI) as assigned to AIS equipment
TacticalPicture:: ClassificationDescriptor	name	N/A	N/A	The name of the entity (e.g. vessel or aircraft)
TacticalPicture:: ClassificationDescriptor	callsign	N/A	N/A	A call-sign used by the entity being classified
TacticalPicture:: ClassificationDescriptor	iso.3166	ISO 3166	2 letter code	Values are the 2 letter code for the country associated with the entity
TacticalPicture:: ClassificationDescriptor	s5516.nat	STANAG 5516 Ed 6	DFI 748 DUI 001	Nationality. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	s5516.nat.ex	STANAG 5516 Ed 6	DFI 748 DUI 003	Extended Nationality. Values are the string representation of the DI bit code
TacticalPicture:: ClassificationDescriptor	icao.fi	ICAO	N/A	Flight Id. Values are the string representation of the aircraft flight id.
TacticalPicture:: ClassificationDescriptor	icao.id	ICAO	N/A	Values are the string representation of the ICAO unique identifier for the aircraft.
TacticalPicture:: ClassificationDescriptor	jc3iedm.air	JC3IEDM v3.1.4	aircraft-type-category-code	Values are the capitalized abbreviations in the Physical Value column
TacticalPicture:: ClassificationDescriptor	jc3iedm.surf	JC3IEDM v3.1.4	surface-vessel-type-category-code	Values are the capitalized abbreviations in the Physical Value column

Extensible Enumeration	Schema Prefix	Reference Specification	Reference Definition	Notes
TacticalPicture:: ClassificationDescriptor	jc3iedm.sub	JC3IEDM v3.1.4	subsurface-vessel-type-category-code	Values are the capitalized abbreviations in the Physical Value column
TacticalPicture:: ClassificationDescriptor	jc3iedm.veh	JC3IEDM v3.1.4	vehicle-type-category-code	Values are the capitalized abbreviations in the Physical Value column
TacticalPicture:: ClassificationDescriptor	app6b.sp	APP-6(B) June 2008	table B-III	Space Entities. Values are the specific one character code within the function id that is associated with the classification
TacticalPicture:: ClassificationDescriptor	app6b.air	APP-6(B) June 2008	table B-IV	Air Entities. Values are the specific one to four character code within the function id that is associated with the classification
TacticalPicture:: ClassificationDescriptor	app6b.ground	APP-6(B) June 2008	table B-V	Ground Entities. Values are the specific one to six character code within the function id that is associated with the classification
TacticalPicture:: ClassificationDescriptor	app6b.surf	APP-6(B) June 2008	table B-VI	Sea Surface Entities. Values are the specific one to four character code within the function id that is associated with the classification
TacticalPicture:: ClassificationDescriptor	app6b.sub	APP-6(B) June 2008	table B-VII	Sea Subsurface Entities. Values are the specific one to four character code within the function id that is associated with the classification
TacticalPicture:: ClassificationDescriptor	app6b.sof	APP-6(B) June 2008	table B-VIII	Special Operations Force Entities. Values are the specific one to four character code within the function id that is associated with the classification
TacticalPicture:: ClassificationDescriptor	app6c.air.icon	APP-6(C) May 2011	air icon	Values are labels from the function column of table 2-4
TacticalPicture:: ClassificationDescriptor	app6c.air.mod	APP-6(C) May 2011	air modifier	Values are from the modifier column of tables 2-5 & 2-7
TacticalPicture:: ClassificationDescriptor	app6c.mis.mod	APP-6(C) May 2011	missile modifier	Values are a concatenation of the modifier column of tables 2-9 & 2-10
TacticalPicture:: ClassificationDescriptor	app6c.land.icon	APP-6(C) May 2011	land icon	Values are labels from the function column of tables 3-3 & 3-4
TacticalPicture:: ClassificationDescriptor	app6c.land.mod	APP-6(C) May 2011	land modifier	Values labels are from the modifier column of tables 3-5 & 3-6
TacticalPicture:: ClassificationDescriptor	app6c.surf.1	APP-6(C) May 2011	sea surface sector 1 modifier	Values are from the modifier column of table 4-2
TacticalPicture:: ClassificationDescriptor	app6c.surf.2	APP-6(C) May 2011	sea surface sector 2 modifier	Values are from the modifier column of table 4-3

Extensible Enumeration	Schema Prefix	Reference Specification	Reference Definition	Notes
TacticalPicture:: ClassificationDescriptor	app6c.surf.icon	APP-6(C) May 2011	sea surface icon	Values are labels from the description column of table 4-5
TacticalPicture:: ClassificationDescriptor	app6c.sub.1	APP-6(C) May 2011	sea subsurface sector 1 modifier	Values are from the modifier column of table 4-11
TacticalPicture:: ClassificationDescriptor	app6c.sub.2	APP-6(C) May 2011	sea subsurface sector 2 modifier	Values are from the modifier column of table 4-12
TacticalPicture:: ClassificationDescriptor	app6c.sub.icon	APP-6(C) May 2011	sea subsurface icon	Values are labels from the description column of tables 4-14, 15, 16, 17 & 18
TacticalPicture:: ClassificationDescriptor	app6c.sp.icon	APP-6(C) May 2011	space icon	Values are from the description column of tables 5-4 & 5-7
TacticalPicture:: ClassificationDescriptor	app6c.sp.mod	APP-6(C) May 2011	space modifier	Values are from the description column of tables 5-5 & 5-6
TacticalPicture:: EntityStatusDescriptor	s5516.wes	STANAG 5516 Ed 6	DFI 394 DUI 009	Weapon/Engagement Status. Values are the string representation of the DI bit code
TacticalPicture:: ActivityDescriptor	jc3iedm.org	JC3IEDM v3.1.4	organisation-status-operational-status-code	Values are the capitalized abbreviations in the Physical Value column
TacticalPicture:: ActivityDescriptor	jc3iedm.org.q	JC3IEDM v3.1.4	organisation-status-operational-status-qualifier-code	Values are the capitalized abbreviations in the Physical Value column
TacticalPicture:: ActivityDescriptor	jc3iedm.org.fire	JC3IEDM v3.1.4	organisation-status-fire-mode-code	Values are the capitalized abbreviations in the Physical Value column
TacticalPicture:: IdentityDescriptor	s5516	STANAG 5516 Ed 6	DFI 376 DUI 007	Non-exercise identities. Values are the string representation of the DI bit code
TacticalPicture:: IdentityDescriptor	s5516.ex	STANAG 5516 Ed 6	DFI 376 DUI 001	Exercise identities. Values are the string representation of the DI bit code
PlanExecution:: AmmunitionCategory	s5516.mis	STANAG 5516 Ed 6	DFI 1622	Non-exercise identities. Values are the string representation of the DI bit code
PlanExecution:: AmmunitionCategory	jc3iedm.amm	JC3IEDM v3.1.4	ammunition-type-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.fire	JC3IEDM v3.1.4	fire-capability-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.weap	JC3IEDM v3.1.4	weapon-type-category-code	Values are the capitalized abbreviations in the Physical Value column

Extensible Enumeration	Schema Prefix	Reference Specification	Reference Definition	Notes
PlanExecution:: CapabilityCategory	jc3iedm.w.sc	JC3IEDM v3.1.4	weapon-type-subcategory-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.mob	JC3IEDM v3.1.4	mobility-capability-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.eng	JC3IEDM v3.1.4	engineering-capability-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.cargo	JC3IEDM v3.1.4	cargo-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.maint	JC3IEDM v3.1.4	maintenance-capability-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.supp	JC3IEDM v3.1.4	support-capability-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.surv	JC3IEDM v3.1.4	surveillance-capability-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.trans	JC3IEDM v3.1.4	transmission-capability-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: CapabilityCategory	jc3iedm.op	JC3IEDM v3.1.4	operational-capability-category-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: DependencyCategory	jc3iedm.mob.dc	JC3IEDM v3.1.4	mobility-capability-descriptor-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: ExtendedPlanStatus	jc3iedm.dev	JC3IEDM v3.1.4	plan-status-development-status-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: ExtendedPlanStatus	jc3iedm.state	JC3IEDM v3.1.4	plan-status-state-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: ObjectiveCategory	jc3iedm.qual	JC3IEDM v3.1.4	action-objective-qualifier-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: OrbitCategory	app6c.sp.mod	APP-6(C) May 2011	space modifier	Values are from the description column of table 5-5
PlanExecution:: ReadinessDescriptor	jc3iedm.org	JC3IEDM v3.1.4	organisation-status-readiness-code	Values are the capitalized abbreviations in the Physical Value column
PlanExecution:: SpecificationDescriptor	jc3iedm.elec	JC3IEDM v3.1.4	electronic-equipment-type-category-code	Values are the capitalized abbreviations in the Physical Value column

Extensible Enumeration	**Schema Prefix**	**Reference Specification**	**Reference Definition**	**Notes**
PlanExecution:: SpecificationDescriptor	jc3iedm.elec.ex	JC3IEDM v3.1.4	electronic-equipment-type-subcategory-code	Values are the capitalized abbreviations in the Physical Value column

Note: that the following Extensible Enumerations can (also) use values from the corresponding Extensible Enumerations with mappings defined above.

- PlanExecution::DependencyCategory : PlanExecution::CapabilityCategory
- PlanExecution::IntentDescriptor : TacticalPicture::ActivityDescriptor
- PlanExecution::ObjectiveCategory : TacticalPicture::ActivityDescriptor
- PlanExecution::PlanType : TacticalPicture::ActivityDescriptor
- PlanExecution::ResourceCategory : TacticalPicture::ClassificationDescriptor
- PlanExecution::ResourceCategory : TacticalPicture::CapabilityCategory
- PlanExecution::TaskingActivity: TacticalPicture::ActivityDescriptor

Note: The SOPES specification provides a UML wrapper for the attributes defined by JC3IEDM.

Implementations use the getSupportMapping method to get a URL to a file to determine a components support for specific Extensible Enumeration values. The file is formatted using JSON as per this non-normative example, which shows how values from the external specifications are appended to the schema prefix.

```
{
        “mapping category”: {
                “descriptor”: “TacticalPicture::ClassificationDescriptor”,
                “values”: {
                        {
                                “value”: “s5516.air.pl.4”,
                                “description”: “BOMBER”
                        },
                        {
                                “value”: “s5516.air.pl.13”,
                                “description”: “MISSILE”
                        },
                        {
                                “value”: “s5516.air.pl.22”,
                                “description”: “CIVIL, AIRLINER”
                        },
                        {
                                “value”: “jc3iedm.air.AIRRW”,
                        "description": "A machine or device capable of
atmospheric flight and dependent on rotating blades for lift."
                  },
                  {
                        "value": "jc3iedm.air. LGTAIR",
                        "description": "A machine or device capable of
atmospheric flight weighing less than the air it displaces."
                  }
            }
      }
}
```

The ResponseExplanation datatype may take values with meanings from Table 13.2 below.

Table 13.2 – ResponseExplanation Extensible Enumeration Mapping

Value	Meaning
tdai.error.none	No error
tdai.deferred.user	Recommendation referred to an operator
tdai.error.noentity	Entity does not exist
tdai.error.noplan	Plan constituent does not exist
tdai.error.noresource	Resource instance does not exist
tdai.error.noref	Reference to another instance is invalid
tdai.error.rule	Recommendation violates a system rule
tdai.error.state	Recommendation is invalid in the current system state
tdai.error.timeout	Recommendation has been timed-out
tdai.error.internal	Unspecified internal error

Implementation specific specializations of these response values are defined by appending a dot ('.') and a specific descriptive string.