TM Forum Technical Report

Intent Guarantee - Intent Extension Model

TR291H

Maturity Level: **General Availability (GA)**	Team Approved Date: 04-Jul-2024
Release Status: Production	Approval Status: TM Forum Approved
Version 3.6.0	IPR Mode: RAND

Notice

of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

Direct inquiries to the TM Forum office:

181 New Road, Suite 304
Parsippany, NJ 07054, USA
Tel No. +1 862 227 1648
TM Forum Web Page: www.tmforum.org

Table of Contents

Executive Summary

The Intent Guarantee model extends the vocabulary of the TM Forum Intent Ontology (TIO) to support the request for, and negotiation of, explicit guarantees. This is the foundation for SLA assurance.

Introduction

In its basic form an intent contain one or more requirements (expectation and conditions) that shall be ensured for the intent to be considered compliant. The requirements are evaluated against the state of the system at the time of reception of the request (this is true regardless of validity expressions). No explicit guarantees are given in terms of prioritizing or ensuring the requirements over time, in their pure form, and as such requirements can be considered best effort goals. The system may find a solution for fulfilling the requirements at the time of acceptance but may later fail to do so, it may be over-allocated, a failure may occur and so on. The requirements are evaluated against absolute constraints, a condition may for instance define an acceptable range for latency, in terms of an absolute min/max value, which will be evaluated against the latest value of a metric. The outcome is a boolean truth value, reflecting a momentary statement of compliance, without tolerance.

The Intent Guarantee model extends the vocabulary of the TM Forum Intent Ontology (TIO) to support the request for guarantees that are future looking and that are statements of enduring compliance. Supporting guarantees implies that an intent handler shall be capable of predicting future outcomes and ensuring requirements with agreed *confidence/probability*, throughout an agreed *guarantee period*, by utilizing methods such as statistical prediction modeling, digital twin simulations, isolation and partitioning of resources, redundant resources, prioritization and so on. The confidence by which the intent handler is expected to issue guarantees is specified as a confidence/probability level by the intent owner and defines the number of outcomes of a requirement that must be positive over time. The exact handling of this is implementation specific, but generally the method used for prediction shall be verifiable using the methods provided by a system for collecting and presenting metrics for instance. Guarantees are also subject to validity expressions, which allows the intent owner to define conditions for when guarantees shall be employed, such as restricting guarantees to certain times, applying different guarantees at different times or requesting guarantees if some other condition is true for instance.

Guaranteed requirements have higher priority that non-guaranteed requirements within an intent and between intents set by the same intent owner. There's no inherent priority between guaranteed requirements emanating from different intent owners, neither is there a priority between different guaranteed requirements defined by the same intent owner. Other extension models may provide such facilities.

A guarantee is specified by associating a *guarantee context* with a requirement (which includes the intent itself, an expectation or a condition), which essentially tells the intent handler that the requirement is to be prioritized over non-guaranteed requirements, and that prediction and long term planning is required. Associating a guarantee context with a requirement does not affect the outcome of the requirement, dependent requirements or the intent itself or its state. Guarantees have a state and associated events which is separate from that of the related intent. An intent may evaluate to a false outcome while one or more guarantees of the intent are non-compliant or vice versa. Reporting of guarantees and intent handler behavior in relation to guarantees is controlled by guarantee reporting expectations.

A guarantee context can be associated with any number of requirements and a requirement may be associated with any number of guarantee contexts, of which only one may be valid at any point in time.

1. Notation and namespaces

The intent guarantee model depends on the following models:

Model	Prefix	Namespaces	Published by	Purpose in the model
Intent Guarantee	ig	`http://tio.models.tmforum.org/tio/v3.6.0/IntentGuaranteeOntology/`	TM Forum	(This model)
Intent Common Model	icm	`http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/`	TM Forum	General ontology model of intent and intent report expression. This document is part of the intent common model specification.
RDF Version 1.1	`rdf`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	W3C	Providing fundamental modeling basics [rdf11]
RDF Schema 1.1	`rdfs`	`http://www.w3.org/2000/01/rdf-schema#`	W3C	Providing schema for knowledge modeling [rdfs11]
XML Schema	`xsd`	`http://www.w3.org/2001/XMLSchema#`	W3C	Providing data types for literal objects [xsd-1] [xsd-2]
Time Ontology in OWL	t	`http://www.w3.org/2006/time#`	W3C	Expression of date and time [owltime]
Examples	ex	http://www..example.org/	IANA	Reserved domain name for examples

Table 1: Model references

The Intent Guarantee model is based on the Resource Description Framework (RDF) [rdf, rdf_mt, rdf_primer] and the Resource Description Framework Schema (RDFS) [rdfs] published by the World Wide Web Consortium (W3C).

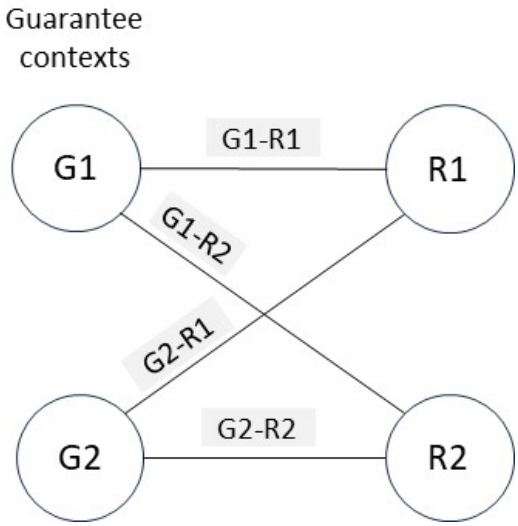
Furthermore, the Intent Guarantee model depends on select models from the TM Forum Intent Ontology such as the intent common model for definition of context and conditions. Further models, such as the Logical Operators Ontology, the Quantity Ontology, the Set Operators model and the Function Definition Ontology can be used to express validity conditions. The Time Ontology in OWL is the base for expressing temporal conditions.

2. General

2.1. Guarantee state machine and events

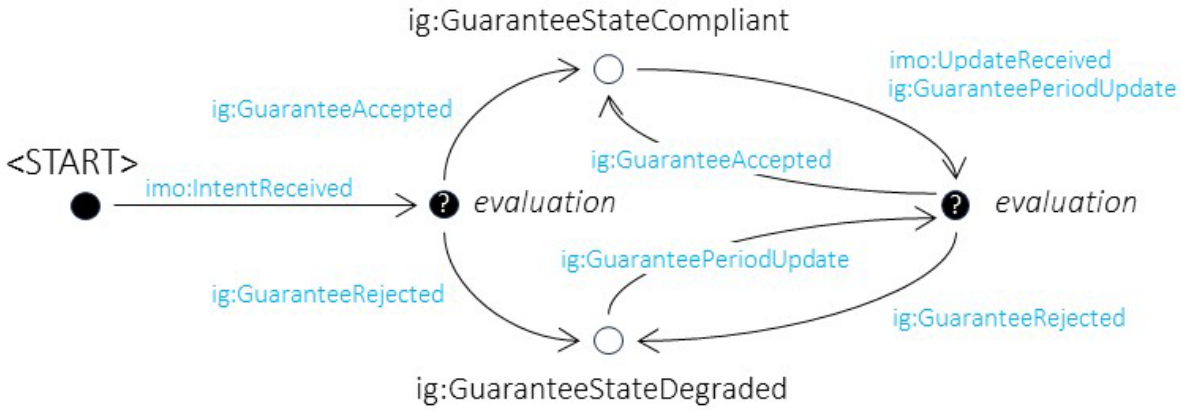
As stated a guarantee has a state, but a guarantee is not a concrete object. It is defined by an association between a requirement (intent, expectation or condition) and a guarantee context. Each association between a guarantee context and a requirement constitutes a unique guarantee.

The picture below shows this concept:



G1 and *G2* are guarantee contexts and *R1* and *R2* are requirements of some kind. The association *G1-R1*, *G1-R2*, *G2-R1* and *G2-R2* are all unique guarantees from a state and evaluation perspective, are also reported separately.

The state machine of a guarantee and related events is depicted below.



The following states are defined:

- `ig:GuaranteeStateCompliant`, the guarantee is momentarily compliant.
- `ig:GuaranteeStateDegraded`, the guarantee is momentarily non-compliant.

Evaluation is a transient state and is not communicated externally.

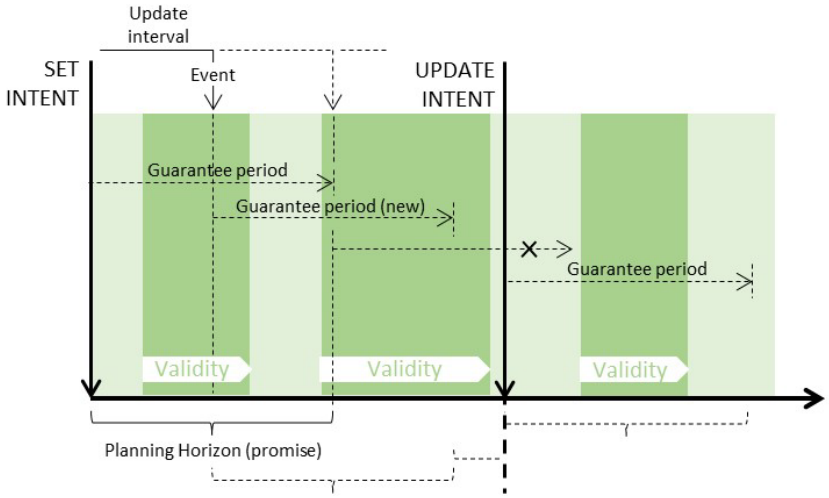
State transitions are triggered by guarantee specific events as well as intent related events. The guarantee specific events are:

- `ig:GuaranteeAccepted`, the intent handler has concluded that it can fulfill the guarantee following an evaluation in preparation for a new guarantee period.
- `ig:GuaranteeRejected`, the intent handler has concluded that it cannot fulfill the guarantee following an evaluation in preparation for a new guarantee period.
- `ig:GuaranteePeriodUpdate`, a new evaluation for the next guarantee period is started, and a report will be triggered when finished.

2.2. Guarantee periods, validity and reporting

The lifespan of a guarantee is tied to the lifespan of the intent and is, by default, expected to be ensured until the intent is removed, and while the associated requirements and the guarantee are valid. Such an indefinite and open-ended approach may be problematic from an evaluation point of view, in terms of computation and ensurance, and may also result in poor resource usage due to excessive over-dimensioning. Similarly, a guarantee, by default, inherits the validity conditions of the associated requirements, which may or may not be the wanted behavior. It is possible to modify this behavior by defining a *guarantee period* and associating the guarantee context with dedicated validity contexts.

- Guarantee period
    - The guarantee period defines the planning horizon for the guarantee and is defined in a *Guarantee Reporting Expectation*. The start of the guarantee period coincides with the time of setting or updating an intent and the guarantee is implicitly renewed (re-evaluated) at the end of the period or at regular intervals, if specified.
- Validity context
    - The guarantee can be given its own validity by association with a validity context. Thus, the validity of the guarantee can be different from the validity of the associated requirements. This can be used to elevate requirements to *guaranteed* during e.g., business hours or specific days of the week, or to modify guarantees over time using multiple guarantee contexts, with different configuration and valid at different times. A guarantee is only applied when valid. Naturally a guarantee is void when the underlying requirement is invalid.



The picture shows an intent and a guarantee that has a number of defined validity periods (during the time depicted). Valid here means that both the guarantee and the associated requirement is valid. The guarantee period is specified as part of a reporting expectation and implies a required minimum planning horizon for the intent handler i.e., the time for which the intent handler shall perform prediction and issue the guarantee. When planning, the intent handler must consider the validity schedule that is applicable within the planning horizon. A guarantee update interval is also specified, which is a trigger for the intent handler to re-evaluate the guarantee for a new guarantee period. If no update interval is specified the guarantee is re-evaluated at the end of the guarantee period. Re-evaluation is a renewed statement of compliance and supersedes a previous statement.

2.3. Guarantee acceptance and negotiation

Guarantees must be evaluated by the intent handler when receiving a new intent, when an intent is updated and when guarantees are renewed. Various outcomes are possible:

- The probability/confidence levels of the associated requirements and guarantee periods can be ensured
- The probability/confidence levels of the associated requirements or guarantee periods can not be ensured
- Neither the probability/confidence levels of associated requirements nor contract period can be ensured

The result of the evaluation is communicated back to the intent owner in the form of *Guarantee Reports* (assuming that one or more appropriate *Guarantee Report Expectations* have been defined). The owner will use the result of a rejection to determine what modifications must be made to the intent for the guarantees to be accepted. The resulting report identifies the guarantee contexts and associated requirements that are non-compliant and the reason(s). Solutions to non-compliance may be to accept shorter guarantee periods, lower probability/confidence levels or relaxing the underlying requirements. There are no new procedures introduced between the intent owner and intent handler due to guarantees. Existing procedures such as setting, updating, probing and judging intents as well as requesting proposals for the best intents are applicable also for guarantees.

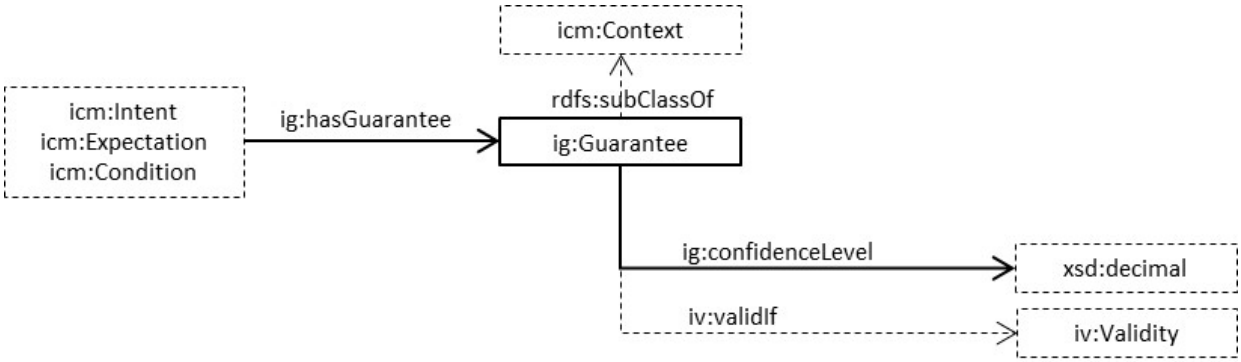
A rejection of a guarantee implies rejection of the intent, if it occurs at the time of setting or modifying the intent. Rejection as a result of a periodic guarantee period update is simply reported, and the state of the guarantee set to `ig:GuaranteeStateDegraded`. The guarantee will be re-evaluated at the end of the next guarantee update period.

3. Requesting a Guarantee

3.1. The guarantee context class

A guarantee is specified by associating an instance of class `ig:Guarantee`, which is a subclass of `icm:Context`, with one or more instances of `icm:Intent`, `icm:Expectation` and `log:Condition` (or subclasses thereof).

The structure of a guarantee context:



The context consists of a description of the guarantee, currently a confidence level (`ig:confidenceLevel`) and validity contexts, both of which as optional. If no confidence level is specified, it is assumed to be 100%, meaning that the intent handler must plan to ensure absolute compliance to requirements. If no validity is specified the guarantee is always valid (for valid requirements).

The confidence level is expressed as a decimal value between 0 and 1 (corresponding to a percentage value).

Guarantee contexts are associated with requirements by an `ig:hasGuarantee` property.

3.2. Integration of guarantees with intents

Example of integration of a guarantee at the intent level:

```
ex:I1 a icm:Intent ;
  ig:hasGuarantee ex:G1  <...>.
ex:G1 a ig:Guarantee ;
  ig:confidenceLevel "0.95"^^xsd:decimal.
```

This example shows a guarantee context *ex:G1* which is associated with the intent *ex:I1* by property `ig:hasGuarantee`. The entire intent and all requirements within it must in this case be guaranteed to be compliant with a probability of 95% as stated by `ig:confidenceLevel`.

A guarantee can similarly be associated with expectations and conditions, for example:

```
ex:I1 a icm:Intent ;
  log:allOf ( ex:E1 ex:E2 )
.
```

```
ex:G1 a ig:Guarantee ;
  ig:confidenceLevel "0.95"^^xsd:decimal.

ex:E1 a icm:PropertyExpectation ;
 ig:hasGuarantee ex:G1 ;
<...>
.

ex:C1 a log:Condition ;
 ig:hasGuarantee ex:G1 ;
<...>
.

ex:C2 a log:Condition ;
<...>
.

ex:E2 a icm:PropertyExpectation ;
 log:allOf ( ex:C1 ex:C2 )
<...>
.

<...>
```

In this case the guarantee context *ex:G1* is associated with expectation *ex:E1* as well as condition *ex:C1*. Condition *ex:C2* is a non-guaranteed requirement. This means that conditions specified within *ex:E1* and condition *ex:C1* has priority over *ex:C2* when choosing and prioritizing between solutions.

It shall be noted here that a single guarantee context can be associated with multiple requirements, from a modeling point of view, but evaluation of the guarantee is done per associated requirement. This implies that the guarantee has a compliance state per such association. Note also that guarantees are reported separately for each association.

3.3. Guarantees with explicit validity

By default, the guarantee validity is implicitly inherited from the associated requirement i.e., if a guarantee is associated with a requirement that has a validity, the guarantee has the same validity. The guarantee can be given its own validity by association with an instance of `iv:Validity` to e.g., elevate requirements to *guaranteed* only during business hours or on specific days of the week for instance, or if other conditions are true. It can also be used to modify guarantees over time e.g., to have different guarantee context valid during different periods.

Example:

```
ex:E1
 a icm:PropertyExpectation ;
 ig:hasGuarantee ex:G1 ;
```

```
 <...>
.

ex:V1
  a iv:Validity ;
  log:if ( imo:Now
           t:inside
           [ t:hasBeginning [ t:inXSDDateTimeStamp "2024-07-
05T17:00+01:00"^^xsd:dateTimeStamp ] ;
             t:hasXSDDuration "PT1H"^^xsd:duration ]
         )
.

ex:G1 a ig:Guarantee ;
  iv:validIf ex:V1
  <...>
.
```

In the example the expectation *ex:E1* is only requested to be guaranteed (it is elevated to guaranteed) from 17:00 (GMT+1) on the 5th of July 2024 and for 1 hour. Outside this time window *ex:E1* is a non-guaranteed requirement. An intent handler must consider the temporal validity of all guaranteed requirements, for the duration of the stipulated contract periods, and ensure that the required resources (within its autonomous domain) are never over-allocated.

4. Guarantee Reporting

4.1. Guarantee reporting expectations

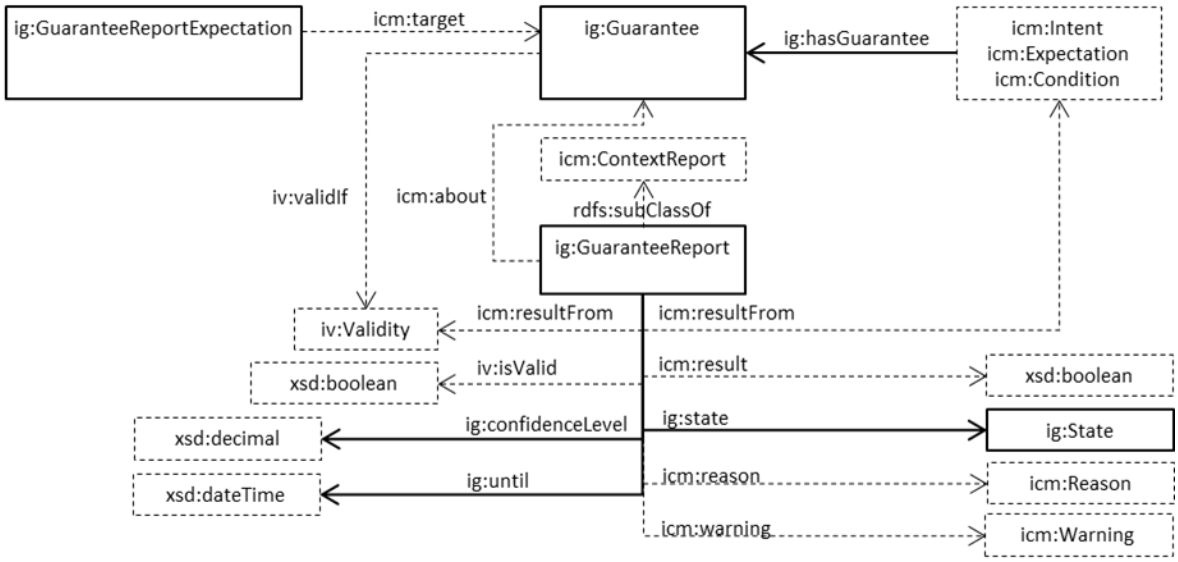
Without a defined guarantee reporting expectation the guarantee period is the same as the intent lifespan, and no reporting with regard to the guarantee will occur. A guarantee reporting expectation of type `ig:GuaranteeReportingExpectation`, which is a subclass of `icm:ReportingExpectation`, allows an intent owner to control the behavior of the intent handler with regard to how guarantees are evaluated and reported.

```
ex:I1 a icm:Intent ;
  log:allOf( ex:E1 ex:RE1 ex:GRE1 ) .
ex:E1 a icm:PropertyExpectation ;
  log:allOf( ex:C1 ) ;
  ig:hasGuarantee ex:G1.
ex:C1 a log:Condition ;  <...>.
ex:GT1 a icm:Target ;  rdfs:member ex:G1.
ex:G1 a ig:Guarantee ;  ig:confidenceLevel "0.95"^^xsd:decimal.
ex:ReportEvent1 a rdfs:Class ;
  rdfs:subClassOf imo:Event ;
  log:match ( [ t:after [ imo:timeOfLastEvent ( ex:ReportEvent1
ex:Intent1 ) ] ;
                t:hasDuration "PT1H"^^xsd:duration
              ]
              t:before
              [ t:hasBeginning imo:Now ]
            ) ;
  imo:eventFor ex:I1
.
ex:GRE1 a ig:GuaranteeReportingExpectation ;
  icm:target ex:GT1 ;  ig:guaranteePeriod "P1DT2H"^^xsd:duration ;
  ig:guaranteePeriodUpdateInterval "PT1H"^^xsd:duration ;
 icm:reportTriggers [ a rdfs:Container ;
              rdfs:member ex:ReportEvent1 ;
              rdfs:member ig:GuaranteeAccepted ;
              rdfs:member ig:GuaranteeRejected ;
              rdfs:member ig:GuaranteePeriodUpdate]
.
```

In the example a property expectation *ex:E1* has an associated guarantee *ex:G1*. A guarantee reporting expectation *ex:GRE1* is defined, which is targeting the guarantee context via target specification *ex:GT1*. The guarantee reporting expectation defines a guarantee period for the context of (`ig:guaranteePeriod`) of 1 day and 2 hours and an update interval (`ig:guaranteePeriodUpdateInterval`) of 1 hour. This tells the intent handler to plan 1 day and 2 hours ahead, every 1 hour. The triggers for guarantee reporting is the periodic event *ex:ReportEvent1* and conditional events for when the guarantee is *accepted*, *rejected and updated*.

4.2. Guarantee reports

The structure of guarantee reports is shown in the figure below:



Guarantees are reported by instances of class `ig:GuaranteeReport`, which is a subclass of `icm:ContextReport`. If a guarantee context is associated with more than one requirement, a separate report will be sent for each such association when triggered. Guarantee reports have different content depending on the trigger (event). The following properties shall always be included:

- Identification of the guarantee context that the report is related to, `icm:about`.
- Identification of the requirement that the report is related to, `icm:resultsFrom`.
- If the guarantee context has a validity the property `iv:isValid` is included and the validity object is identified with an `icm:resultsFrom` property

The rest of the content is optional and depending on the trigger (the event indicated in the intent report), intent handler capabilities and other conditions:

- Event `ig:GuaranteeAccepted`
    - This trigger is applicable when *setting*, *probing*, *judging* and *updating* an intent and when the guarantee is renewed as a result of an interval update.
    - The intent handler has concluded that it can issue the guarantee.
    - The predicted boolean outcome of the next guarantee period, `icm:result`, is *true (compliant)*.
    - The predicted confidence level of the guarantee for the next guarantee period, `ig:confidenceLevel`, has a value of at least the requested.
    - The predicted guarantee state, `ig:state`, is set to `ig:GuaranteeStateCompliant`.
    - The end of the guarantee period, `ig:until`, is set.

- Event `ig:GuaranteeRejected`
    - This trigger is applicable to the same procedures as `ig:GuaranteeAccepted`.
    - The intent handler has concluded that it cannot issue the guarantee.
    - The predicted boolean outcome of the next guarantee period, `icm:result`, is *false (non-compliant)*.
    - The predicted confidence level of the guarantee for the next guarantee period, `ig:confidenceLevel`, has a value of less than the requested.
    - The predicted guarantee state, `ig:state`, is set to `ig:GuaranteeStateDegraded`.
    - The end of the guarantee period, `ig:until`, is set.
    - Zero or more reasons for the failure, property `icm:reason`, may be included.
        - This is information that can guide the intent owner to modify the guarantee in a way that may be obtainable.
    - Zero or more warnings, property `icm:warning`, may be included.
        - This is information that the intent owner shall be made aware of, but that are not reasons for failure, such as providing configuration parameters that are valid, but that may not be what was intended.
- Event `ig:GuaranteePeriodUpdate`
    - Indicates that evaluation has started and is ongoing for the next guarantee period.
    - The report will be followed by a new report with results, if requested in the reporting expectation, when the evaluation and planning phase for the next guarantee period is completed.

The following reasons, which are subclasses of `icm:Reason`, are defined for guarantee reports:

- `ig:GuaranteePeriodTooLong`, the specified guarantee period is too long for the intent handler to accurately predict and plan for the guarantee, this is a result of limitations in implementation.
- `ig:GuaranteePeriodNotObtainable`, the intent handler is unable to issue a guarantee for the requested duration.
- `ig:ConfidenceLevelNotObtainable`, the intent handler is unable to issue a guarantee with the requested confidence level for the guarantee period.
- `ig:ConfidenceLevelNotReached`, the measured confidence during the evaluation window is lower than requested.
- `ig:ConflictingGuarantees`, a requirement is associated with more than one guarantee context, of which more than one is valid at some point in time during the upcoming guarantee period.

The following warnings, which are subclasses of `icm:Warning`, are defined for guarantee reports:

- `ig:GuaranteePeriodUpdateIntervalTooLong`, the interval is longer than the guarantee period. The result is that the interval is void and re-evaluation will occur at the end of the guarantee period.

An example of reports issued as a result of a rejection of one condition (*ex:C1*) in an intent with two guaranteed conditions (*ex:C1* and *ex:C2*), associated with the same guarantee context *ex:G1*:

```
ex:IR1 a icm:IntentReport ;
  icm:event [ a ig:GuaranteeRejected ] ;
icm:event [ a ig:GuaranteeAccepted ] ;
<...>
.

<...>
ex:GR1 a ig:GuaranteeReport ;
  ig:state [ a ig:GuaranteeStateDegraded ]
  icm:about ex:G1 ;
  icm:resultFrom ex:C1 ;
  icm:result false ;
  icm:reason [ a ig:ConfidenceLevelNotObtainable ] ;
  ig:confidenceLevel "0.8"^^xsd:decimal ;
 ig:until "2024-04-26T14:30:00+01:00"^^xsd:dateTime
.
ex:GR2 a ig:GuaranteeReport ;
  ig:state [ a ig:GuaranteeStateCompliant ]
  icm:about ex:G1 ;
  icm:resultFrom ex:C2 ;
  icm:result true;
   ig:confidenceLevel "0.97"^^xsd:decimal ;
ig:until "2024-04-26T14:30:00+01:00"^^xsd:dateTime
.
```

In the example the first report, about *ex:C1*, indicates that whatever confidence level that was requested cannot be obtained, given the defined guarantee period. The predicted confidence level is 0.8 (80%) from the time of the report until 2024-04-26 at 14:30 (CET). The second report, about *ex:C1*, indicates that the condition can be guaranteed with at least the confidence requested and for the required guarantee period. The confidence level reported is equal to or higher than requested and the promise is again valid until 2024-04-26 at 14:30 (CET).

As one guarantee is non-compliant the intent would be rejected, if part of an operation that sets or modifies the event.

4.3. Integration of guarantee reports with other reports

If a guarantee reporting expectation is defined any issued guarantee report will be associated with an intent, expectation or condition report by property `ig:guaranteeReport`, which is a subclass of `icm:contextReport`.

Example:

```
ex:IR1 a icm:IntentReport ;
  icm:event [ a imo:Degrades ] ;
  icm:about ex:I1 ;
icm:result true ;
icm:resultFrom ex:ER1
.
ex:ER1 a icm:PropertyExpectationReport ;
icm:about ex:E1 ;
icm:result true ;
icm:resultFrom <...> ;
ig:guaranteeReport ex:GR1
.
ex:GR1 a ig:GuaranteeReport ;
  ig:state [ a ig:GuaranteeStateCompliant ]
  icm:about ex:G1 ;
  icm:resultFrom ex:ER1 ;
  icm:result true ;
   ig:confidenceLevel "0.95"^^xsd:decimal
ig:until "2024-04-26T14:30:00+01:00"^^xsd:dateTime
.
```

In the example an intent report *ex:IR1* is issued, which takes results from *ex:ER1*, which is a guaranteed property expectation. The associated guarantee report *ex:GR1* is referred to with property `ig:guaranteeReport`. The reason for the report is a degradation of the intent, but the predicted guarantee is compliant. The predicted confidence level is 0.95 (95%) until 2024-04-26 at 14:30 (CET).

5. Administrative Appendix

5.1. Document History

5.1.1. Version History

Version Number	Date Modified	Modified by:	Description of changes
0.1	26-Apr-2024	Dan Green	Initial draft
3.5.0	03-May-2024	Alan Pope	Final edits prior to publication
3.6.0	04-Jul-2024	Alan Pope	Final edits prior to publication

5.1.2. Release History

Release Status	Date Modified	Modified by:	Description of changes
Pre-production	03-May-2024	Alan Pope	Initial Release
Pre-production	10-Jun-2024	Adrienne Walcott	Updated to Member Evaluated status
Pre-production	03-May-2024	Alan Pope	Updated to v3.6.0
Production	30-Aug-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status

5.2. Acknowledgments

Team Member (@mention)	Company	Role*
Jörg Niemöller	Ericsson	Project Co-Chair
Dan Gren	Ericsson	Author
Kevin McDonnell	Huawei	Project Co-Chair
Yuval Stein	Amdocs	Project Co-Chair
Kamal Maghsoudlou	Ericsson	Key Contributor
Leonid Mokrushin	Ericsson	Key Contributor
Marin Orlić	Ercisson	Key Contributor
Alan Pope	TM Forum	Additional Input
Dave Milham	TM Forum	Additional Input
Xiao Hongmei	Inspur	Reviewer

**Select from: Project Chair, Project Co-Chair, Author, Editor, Key Contributor, Additional Input, Reviewer*

6. Appendix A: Vocabulary Reference

This chapter contains a reference definition of all model vocabulary. It is sorted alphabetically.

6.1. GuaranteeAccepted

The class `ig:GuaranteeAccepted` specifies an event. The meaning of the event is that the intent handler has concluded that it is capable of issuing a requested guarantee for the next guarantee period. The guarantee target state is `GuaranteeStateCompliant`.

Instance of: `rdfs:Class`
Sub-class of: `imo:Event`

6.2. confidenceLevel

The property `ig:confidenceLevel` specifies a required, predicted or real (measured) confidence/probability level, as a value between 0-1 (percentage). If not specified the confidence is assumed to be 1 (100%).

Instance of: `rdf:Property`
Domain: `ig:Guarantee, ig:GuaranteeReport`
Range: `xsd:decimal`

6.3. ConfidenceLevelNotObtainable

The class `ig:Complies` specifies a reason for rejection of a guarantee. The requested confidence level is not possible to reach for during the guarantee period, according to predictions.

Instance of: `rdfs:Class`
Sub-class of: `icm:Reason`

6.4. ConfidenceLevelNotReached

The class `ig:Complies` specifies a reason for degradation of a guarantee. The requested confidence level was not reached during the evaluation window.

Instance of: `rdfs:Class`
Sub-class of: `icm:Reason`

6.5. ConflictingGuarantees

The class `ig:Complies` specifies a reason for rejection of a guarantee. A requirement is associated with more than one guarantee context, of which more than one is valid at some point in time during the upcoming guarantee period.

Instance of: `rdfs:Class`
Sub-class of: `icm:Reason`

6.6. Guarantee

The class `Guarantee` is a context that, when associated with requirements (objects of type `icm:Expectation`, `icm:Condition` and `icm:Intent`), modifies the status of those requirements to "guaranteed".

Instance of: `rdfs:Class`
Sub-class of: `icm:Context`

6.7. guaranteePeriod

The property `ig:guaranteePeriod` specifies the duration of guarantee periods, in ISO8601 format. If not specified in a guarantee reporting expectation the guarantee period is the entire life-span of the associated intent, which is generally indefinite unless limited by other vocabulary.

Instance of: `rdf:Property`
Domain: `GuaranteeReportingExpectation`
Range: `xsd:duration`

6.8. GuaranteePeriodUpdate

The class `ig:GuaranteePeriodUpdate` specifies an event. The meaning of the event is that a new evaluation and planning phase has been initiated as a result of periodic updating or the end of the previous guarantee period. The guarantee target state is either `GuaranteeStateCompliant` or `GuaranteeStateDegraded`, depending on the outcome.

Instance of: `rdfs:Class`
Sub-class of: `imo:Event`

6.9. guaranteePeriodUpdateInterval

The property `ig:guaranteePeriodUpdateInterval` specifies the interval of re-evaluation and planning of guarantee periods, in ISO8601 format. If not specified in a guarantee reporting expectation the guarantee period update interval is the same as the guarantee period.

Instance of: `rdf:Property`
Domain: `ig:GuaranteeReportingExpectation`
Range: `xsd:duration`

6.10. GuaranteePeriodNotObtainable

The class `ig:Complies` specifies a reason for rejection of a guarantee. It is not possible to guarantee the requirement during the requested guarantee period.

Instance of: `rdfs:Class`
Sub-class of: `icm:Reason`

6.11. GuaranteePeriodTooLong

The class `ig:Complies` specifies an reason for rejection of a guarantee. The requested guarantee period is too long for the intent handler to accurately predict and plan for the guarantee, this is a result of limitations in implementation.

Instance of: `rdfs:Class`
Sub-class of: `icm:Reason`

6.12. GuaranteePeriodUpdateIntervalTooLong

The class `ig:GuaranteePeriodUpdateIntervalTooLong` specifies a warning. The requested guarantee period update interval is longer than the guarantee period. The result is that it is void and the guarantee period and update interval is the same.

Instance of: `rdfs:Class`
Sub-class of: `icm:Reason`

6.13. GuaranteeRejected

The class `ig:GuaranteeRejected` specifies an event. The meaning of the event is that the intent handler has concluded that it is not capable of issuing a requested guarantee. The guarantee target state is `ig:GuaranteeStateRejected`.

Instance of: `rdfs:Class`
Sub-class of: `icm:Event`

6.14. GuaranteeReportingExpectation

The class `ig:GuaranteeReportingExpectation` specifies a reporting expectation for guarantees and is used to configure a specific guarantee period and conditions for when to issue reports related to guarantees

Instance of: `rdfs:Class`
Sub-class of: `icm:ReportingExpectation`

6.15. GuaranteeStateCompliant

The class `ig:GuaratneeStateCompliant` specifies a state. The meaning of the state is that the intent handler has concluded that it's capable of delivering the guarantee for the next guarantee period. Possible events transitioning from this state are `imo:UpdateReceived and ig:GuaranteePeriodUpdate.`

Instance of: `rdfs:Class`
Sub-class of: `ig:State`

6.16. GuaranteeStateDegraded

The class `ig:GuaratneeStateDegraded` specifies a state. The meaning of the state is that the intent handler has concluded that it's not capable of delivering the guarantee for the next guarantee period. Possible events transitioning from this state are `imo:UpdateReceived and ig:GuaranteePeriodUpdate.`

Instance of: `rdfs:Class`
Sub-class of: `State`

6.17. hasGuarantee

The property `ig:hasGuarantee` is used to associate a guarantee context with a requirement (objects of type `icm:Expectation`, `icm:Condition` and `icm:Intent`).

Instance of: `rdf:Property`
Domain: `icm:Expectation, icm:Condition, icm:Intent`
Range: `ig:Guarantee`

6.18. State

The class `ig:State` is a base-class of Guarantee states.

Instance of: `rdfs:Class`

6.19. state

The property `ig:state` specifies the state of a guarantee.

Instance of: `rdf:Property`
Domain: `ig:GuaranteeReport`
Range: `ig:State`

6.20. until

The property `ig:until` specifies the expiry date and time of a guarantee period, in ISO8601 format.

Instance of: `rdf:Property`
Domain: `ig:GuaranteeReport`
Range: `xsd:dateTime`

6.21. Vocabulary

The object `ig:Vocabulary` is a container of all model elements.

Instance of: `rdfs:Container`