The Finnish national body votes YES for the progression of ISO/IEC DIS 13235 –– ODP Trading function with the following comments:

Item: 1
Clause: 6, page 5 lines 17–21
Classification: E
Rationale: Concept of policy is unclear and misleading.
Suggestion:

Replace
"A policy is a set of rules with a particular objective."
by
A policy is a set of rules with a particular objective, or in other words, a description of the (collective) behavior that is characteristic to the object(s) whose policy/ies are considered.

Replace
"The policies identified provide a framework within which to formulate the configurable aspects of a trader object's behavior."
by
"The policies identified provide a framework within which the trader object's behavior may be implemented or configured."

Item: 2
Clause: 6 (lines 7–8), 6.2.4, 6.2.6
Classification: E
Rationale: Inconsistency in the text.
Suggestion: Add the missing roles to clause 6.

Item: 3
Clause: 6.4.3
Classification: E
Rationale:

It is very confusing, that the arbitration policy shows that exports are probably propagated, but the export policy does not reveal this at all.

Suggestion: Remove the first example.

Item: 4
Clause: 8 (example case 8.2.1)
Classification: E

Rationale:

This document makes the relationship between CORBA and ODP work unclear. The CORBA platform is throughout chapter 8 treated as a special case: the decision on representation of concepts in CORBA is prescribed while other platforms are allowed to do their decision independently. Generally, in CORBA, concepts have been realized through language binding standards, separately from the concepts definition. However, this document, that is not even a CORBA document, defines things that should like−wisely be a separate binding document of OMG.

We are also concerned about the readability of the IDL descriptions in chapter 8. The readers of this document get confused of the amount of cases used. The IDL description would be more informatic with less intermediate concepts (structure and list names).

As we do not have resources at the moment for writing the whole IDL description we only give some examples on how we think that the description should look like. The whole of chapter 8 should be simplified with similar rules.

The COS description in Annex B can be used as a style guide.

Suggestion:

Work through chapter 8 with similar changes as described in the following example with **clause** 8.2.1.:

Start the **clause** by describing what the Interface Identifiers are for.

Replace contents of box with

```
typedef any InterfaceIdentifierType;
typedef sequence <InterfaceIdentifierType> InterfaceIdListType;
```

Add note that explains that platforms like CORBA should declare standard binding for InterfaceIdentifierType themselves.

Item: 5
Clause: 8 (example case 8.2.6, page 25)
Classification: E
Rationale:

The IDL descriptions try to specify not only the syntax but also the semantics. The english text should be used for describing extra semantic restrictions, and the IDL description should allow more. With a simpler IDL description, the standard is more readable and also guides to better implementations.

Similar structures appear throughout chapter 8 and should be corrected as the example given for the page 25 of **clause** 8.2.6.

Suggestion:

Replace the first box on page 25 with the following:

```
enum OperandKind {PROPERTY, LITERAL};
enum SimpleRuleIndicatorType {EXISTS, COMPARE};

union OperandType
switch (OperandKind) {
case PROPERTY:
      TaggedPropertyNameType taggedPropertyName;
case LITERAL:
      any literalValue;
}

struct CompareType {RelationalOperatorType operator;
      OperandType  operand1;
      OperandType  operand2;
}

union SimpleRuleType
switch (SimpleRuleIndicatorType) {
case EXISTS:
      TaggedPropertyNameType taggedPropertyName;
case COMPARE:
      CompareType comparison; };
```

Item: 6
Clause: 8.1.
Classification: e
Rationale: The text seems to be inconsistent.

Suggestion: Simplify the text as follows:

8.1. Correspondence with information viewpoint

The information specification defines a directed graph where partitions of offer space are placed at the nodes of the graph. The only constraint discernible from the information viewpoint is that, for a given direction, only one edge is allowed between any two nodes. Two edges, with opposite directions, are permitted between the same two nodes.

The computational specification defines trader objects, i.e. objects that provide a trading service at an interface. A trader object may use a trading service of another trader, i.e. it is a client of that other trader and thus linked to it.

The relationship between these two viewpoint specifications is prescribed in this standard. An information partition may not be split to several trader objects, however, many partitions are allowed to be associated with a single trader object. For each partition, there is a trading service interface that allows import search to be started at that partition. The partitions within the trader object may be connected by edges and therefore the search may propagate through all partitions within the trader object.

Item: 7
Clause: 8.2.6, page 25
Classification: E
Rationale:

In the second box the IDL specification allows empty lists to appear in notRule, andRule and orRule, but no semantics for those empty lists are given.

Suggestion:

Add below the box:

An instance of RuleConstractType with an empty notRule corresponds to the boolean value true.
An instance of RuleConstractType with an empty andRule corresponds to the boolean value true.
An instance of RuleConstractType with an empty orRule corresponds to the boolean value false.

Item: 8
Clause: 8.2.11
Classification: E
Rationale: See **Item** 4.

Suggestion:

Replace object by InterfaceIdentifierType.

Add semantic description of global and local offers: what they are used for and what information they have to contain.

Item: 9
Clause: 8.3.1.
Classification: E
Rationale:

The concept of exceptions is very familiar from programming languages and many system structures. However, ODP framework does not talk about exceptions, but the operation interfaces have a single invocation signature and multiple termination signatures.

As the trading function standard is the first component standard, it will be used as an example when writing other component standards. Therefore, it is very essential, that the correct ODP terms are used wherever possible.

Suggestion:

Write a short paragraph that explains the relationship between the IDL exception and the variety of termination signatures. Keep the english text talking about terminations, and the IDL about exceptions. The standard can not expect the exception system to be the native concept of all ODP systems.

Item: 10
Clause: 8.4.1.1
Classification:
Rationale:
Suggestion:

Make sure that the granularity of the expiry date values is enough for all application areas. Potential values range from milliseconds to years.

Item: 11
Clause: 8.4, before 8.4.1
Classification: E
Rationale:

The whole chapter 8.4 is very difficult to read and understand. Additional information in the beginning of this chapter would help the reader a lot. The readers are especially missing two things: why the standard properties are classified (partitioning to chapters 8.4.1 – 8.4.5, and what kind of values these properties may carry (data types).

Suggestion:

Add between titles of 8.4. and 8.4.1.:

The standard properties are classified into five classes, namely (names of the classes, see **Item** 12), because each class captures a set of properties that allow cooperation between the trader itself and another objects.

The standard properties have value types such as boolean, small integer values, and rule language expressions.

Item: 12
Clause: 8.4, 8.2.6
Classification: E
Rationale:

In clause 8.2.6., on page 24, one classification of property names is given in order to make five separate name spaces. In clause 8.4. a different classification is used that is almost mappable to the classification in 8.2.6.

We suggest the following mapping between 8.2.6. and 8.4:

A. Standard service properties

– compare with service property name in 8.2.6

– missing from 8.4

B. Standard offer properties

– comp. with offer property name

– standard service offer properties in 8.4

C. Standard link properties

– comp. with link property name in 8.2.6

– standard link properties in 8.4

D. Standard trader properties

– comp. with trader property name in 8.2.6

– standard trader properties
and standard trader properties for policies in 8.4

E. Standard properties

– comp. with standard property name in 8.2.6

– standard property names in 8.4

For locating the properties under the new classes, see **Item** 13.

Suggestion:

Use as a classification in 8.4 the following:

A. Standard service properties
B. Standard offer properties
C. Standard link properties
D. Standard trader properties
E. Standard properties

Item: 13
Clause: 8.4
Classification: E
Rationale:

The structure of this chapter is very difficult to follow, as the same properties appear repeatedly in several classes. Needs reorganization.

Also, many of the properties should belong to more than one class. The suggested table includes changes in this respect.

SOE is not understood by readers.

Suggestion:

Do not repeat the text of the properties when they reappear under different subsections. It might be easier to read the text, if there were
1) explanation of the classification of the properties,
2) one list of properties, together with value types and
3) a table with properties and classes at rows and columns and crosses indicating all classes that a property can be used in.

property	classes
expiry date	offer properties
link name	link and standard properties
type manager interface identifier	link and trader properties
type manager interface for offer properties	trader properties
search time	trader and standard properties
searched cardinality	trader and standard properties
matched cardinality	trader and standard properties
retrieved cardinality	trader and standard properties
offer iterator threshold	trader and standard properties
iterator retention time	trader and standard properties
depth to propagate search	trader and standard properties
trading subgraph	trader properties
type management criteria	trader and standard properties
search criteria	trader and standard properties
offer acceptance criteria	trader and standard properties
offer placement criteria	trader and standard properties
permit to use SOE	trader and standard properties

Replace SOE by "service offer evaluator".

Item: 14
Clause: 8.4
Classification: E
Rationale:

Suggestion:

Remove the following properties:
– abstract signature type supported
– starting offer partition

Item: 15
Clause: 8.4.2.3
Classification: e
Rationale:

Over−specification.

In the previous editing meeting, the agreement was that type manager interface identifier is reserved as the name of an link property with the described meaning. However, it was not intention to require all traders to store this information into all links. This is only a potential method of storing the information for better performance. This information is also only used internally by the trader object itself.

Suggestion: Add a note that tells that storaging the information is optional.

Item: 16
Clause: 8.5.2.4, 8.5.3
Classification: E
Rationale:
Suggestion:

Remove the clause and title of 8.5.2.4. and move the text of that clause to 8.5.3, as it describes the trader objects' behavior in a correct environment.

Item: 17
Clause: 8.6.5.2
Classification: e
Rationale:

The general assumption in ODP framework is to avoid exception situations whenever an operation can be considered successful. What an successful operation execution then means? It means that the system status is after the operation such that it does not contradict to the postconditions of the operation. Therefore, if an offer does not exist, removing it should always succeed.

It is admittable that the situation is suspectable. Extra information can be delivered to the operation invocator as qualification codes, just like in import operation.

Suggestion: Remove word "error" from exceptions.

Item: 18
Clause: 8.6.6.3, 8.6.1
Classification: response to editors request in clause 8.6.6.3
Rationale:

In a distributed system where dynamic values and requests to remote objects are Therefore, semantics for missing values is needed. An optional value is only a form of missing values in general.

We suggest that when the trader is searching things that are required to match to matching criteria, a missing value produces a non−match result, and when the trader is searching things that are preferred to match to preference criteria, a missing value produces a match result.

In the presence of preference criteria, the set of selected offers may include offers that do not match to the preference criteria, but only to the matching criteria. A missing value just says "I don't know whether this matches or not to your preferences" while some other offers may even say "although I now this is not what you preferred").

Suggestion:

Add to **clause** 8.6.1 text that carries the following idea:

When the trader is searching things that are required to match to matching criteria, a missing value produces a non−match result. When the trader is searching things that are preferred to match to preference criteria, a missing value produces a match result.

Reply to question in 8.6.6.3:
The modify offer −operation can not change properties from static to dynamic, or vice versa.

Item: 19
Clause: 8.7.6, 8.7.7
Classification: E
Rationale: Atomicity of operations should be preserved.
Suggestion:

Correct the last sentence to indicate that it is not prescribed whether the result is a before modification or after modification state, but that is should be either of them.

Item: 20
Clause: 9.1
Classification: E
Rationale: Over−specification.
Suggestion:

Remove requirement for a trader object to conform to service offer evaluator interface. If such interface is implemented it should conform, but the existence of such interface is not compulsory.

Item: 21
Clause: Annex A, note in A.1
Classification: E
Rationale:

Gives wrong advise to implementors. There is no requirement that the implementor uses the computational viewpoint description as implementation basis as such. For example, for performance reasons, the implementor may replace things by more static variations.

Suggestion:

Remove temporary note.
Add another note that encourages refinements.