

# Contexts, Views and Rules: An Integrated Approach to Trader Contexts

M.Y. Bearman<sup>a</sup> and K.A. Raymond<sup>b</sup>

<sup>a</sup>myb@ise.canberra.edu.au  
CRC for Distributed Systems Technology  
Faculty of Information Sciences and Engineering  
University of Canberra, Bruce 2616, Australia

<sup>b</sup>kerry@dstc.edu.au  
CRC for Distributed Systems Technology  
Centre for Information Technology Research  
University of Queensland, 4072, Australia

## Abstract

Service offers within a trader can be grouped into contexts. By associating each context with a membership rule, traders can be federated in a transparent manner without compromising heterogeneity and autonomy. In addition, context views can be defined to provide importers with an individualised view of the search offer space.

Keyword Codes: C.2.4

Keywords: Computer-Communication Networks, Distributed Systems

## 1 INTRODUCTION

A trader is an object to which an exporting object can advertise its services and from which an importing object can find its needs from the set of advertised service offers in a distributed environment.

A *service offer* describes a service that is being traded. It is an assertion made by a server about a service that is offered for use by other objects at a computational interface. It consists of:

- a service interface identifier (where to obtain the service),
- a service type identifier (what kind of service)
- values of service properties (detailed descriptions of the service)
- an optional policy interface identifier (a handle for dynamic considerations).

An export operation passes a service offer to the trader.

A *service request* is a request of a client for service offers satisfying some needs expressed in matching constraints. A service request nominates:

- a service type identifier (what kind of service),
- boolean expression on service properties of the nominated service type (matching constraint).

An import operation passes a service request to the trader and returns a set of service offers which satisfy the request.

A collection of autonomous component traders can federate (be linked without loss of autonomy) to achieve:

- a larger choice of advertised service offers for importers
- a wider market of service offers for exporters

ISO and ITU-T (formerly CCITT) are currently developing an international standard on Trader [1].

## 2 MOTIVATION

A trader could contain many service offers. To manage a large set of offers, the trader groups these offers into *contexts*.

A context consists of a group of service offers that are related in some way. The groupings might reflect the requirements of the:

- exporters. For example, service offers could be grouped in contexts that determine the visibility of the service offers to importers for security and accounting reasons.
- importers. For example, service offers could be grouped into contexts that associate similar services to reduce the amount of searching required.
- administrator. For example, service offers could be grouped into contexts that reflect the administration structure or physical structure of storage locations of the service offers.

Any context can be further refined into sub-contexts, creating a context hierarchy. The current trend [2, 3, 4] is to restrict the context hierarchy to be a single shared tree using a common naming scheme (usually context-relative naming). The disadvantages of this approach are:

- it imposes a single structure on all members of the trading community (importers, exporters, and administrators) which is unlikely to satisfy the disparate needs of the community members
- a tree structure cannot reflect that some groupings overlap. For example, if physical location is the basis for partitioning, then it is impossible to have a grouping of services based on functionality.
- it forces users of one trader to learn the context hierarchy of other traders in their federation

### 2.1 Federation

Difficulties emerge when traders with different context hierarchies federate.

The users of a trader (both people and applications) are familiar with their local trader context. When their trader becomes federated with another trader, the other trader might have a context hierarchy structured according to different policies. While a human user might be able to learn the context structure of the other trader and thus make effective use of the enlarged space of service offers, existing applications are unable to learn new context structures and will not be able to take advantage of the federation. Even a human user's ability to learn will be tested by a federation of many traders, all with different context structures. Such a federation cannot be made transparent to its users (an important aim of open distributed processing).

For effective federation, users need to access the service offer space of the federation through the context structure of their local trader. This could be achieved by restricting federation to traders with similar context hierarchies, but such homogeneity is inappropriate for open distributed processing. Alternatively, context hierarchies could be completely standardised, but this is a severe restriction on the autonomy of individual administrations within an ODP system.

This paper proposes a third solution which accommodates heterogeneity, autonomy, and transparency, consistent with the goals of Open Distributed Processing. A trader is free to construct any context hierarchy appropriate to its users. Users import and export through the contexts defined within their local traders. Access to the service offer space of federated traders is achieved transparently; the local trader transforms operations involving locally-defined contexts into operations on federated traders by "explaining" the semantics of the locally-defined contexts to the federated traders.

Since a context is a grouping of related service offers, the semantics of a context is the criteria for membership of that context. For example, the context might contain service offers of the same service type or having a common provider. By associating each context with membership rules, the requested local context can be translated into rules of matching requirements when federating with other traders.

For example, a trader has a context which contains service offers provided by Citibank. The membership rule for the Citibank context could be based on service properties, e.g. "organization = Citibank". If a user wishes to open a Citibank account at a local branch, then the import operation would be

```
import BankTeller from Citibank where branch = Brisbane
```

The local trader could access a remote trader with the following translated request:

```
import BankTeller where organization = Citibank and branch = Brisbane
```

The local trader translates a local context into the context's membership rule and adds it to the original matching constraint to form a matching constraint that can be interpreted by the remote trader. The remote trader searches its context hierarchy using this extended matching constraint on behalf of the local trader in the same way as it searches for any of its own users. Thus, when federating, a trader needs to know only the rules of its own contexts, and neither the trader nor its users need to understand the context structure of the remote trader.

## 2.2 Other Benefits of Rule-based Contexts

In addition to allowing transparent federation, context membership rules have a number of other benefits.

A service offer can satisfy the membership rules of many contexts and thus be a member of many contexts. Membership rules do not enforce a partitioning or tree structure; the contexts can support the simultaneous categorization of a service offer on a variety of criteria. Rule-based context hierarchies can be viewed as (potentially) overlapping sets (like a Venn diagram) rather than a tree. This more flexible approach benefits both the importers and exporters.

The presence of context membership rules can prevent a service offer from being exported into an inappropriate context. This reduces the potential for both error and malice on the part of the exporter.

For importers, there is an even greater benefit. Since a context is just a set of rules, an importer can now express its frequently-used matching rules as contexts. That is, an importer can group the set of service offers into *import context views* that are convenient for its own, individualised use. For example, a keen investor could create an import context view containing all the BankManager interfaces offering an interest rate of over 10% per annum. Import context views are similar to database views in that the import context view does not need to exist within the trader; the trader can use the context rules to generate the import context view as required. The use of import context views will simplify an importer's use of the trader.

## 3 INFORMATION MODEL FOR TRADING

Our proposed information model for context is rule-based. Each context has a membership rule to group service offers into related sets. A rule-based context allows the ready federation of traders with different context hierarchies. Furthermore, an importer can support logically independent groupings of service offers, by providing individualized membership rules to form context views.

### 3.1 Contexts as Rules

A context is a set of service offers grouped by some commonality. Each context has a unique identifier (name) and a membership rule which is the set predicate. Each service offer that is a member of a context must satisfy the context membership rule.

A context can be identified either by its identifier or by its associated predicate. The context identifier is a short-hand notation for the details of the membership rule. If a rule is assigned a context identifier, then future reference to this context only need to nominate the context identifier without repeating the details of the rule. Use of context identifiers in operation specifications reduces redundancy and simplifies operation specification.

A membership rule can use arithmetic, boolean, and set expressions. The operands can be any of:

- context identifiers,
- service type identifiers,
- service properties,

- trading service properties of the trader.

One membership rule can group service offers by service type; another by the service property of provider; another by a rule, such as:

```
{printers and cost < 5 cents_per_page and offer_location = local} ⊆ Citibank_context
```

Relationships can exist between different contexts. These relationships can be determined by analysis of the membership rules or they can be specified at context definition time. The trader can use context relationships to improve search strategies (see Section 5).

### 3.2 Trader contexts

The set of all service offers stored by a trader forms a trader context. A trader context can be further refined into sub-contexts for the convenience of that trader community (importers, exporters, and administrators).

Trader context hierarchies can be used to reflect trader administrative decisions, e.g. storage location of the service offers within the trader. More importantly, trader context hierarchies can be used to implement local enterprise policy, e.g. security considerations might allow access by some importers and exporters to only certain contexts.

The trader context hierarchy is potentially visible to all importers and exporters (apart from any enterprise restrictions). However, the trader might support a number of disparate groups of users who would prefer to have different parts of the trader context hierarchy visible to them. These needs can be accommodated by the use of a number of trader interfaces. Associated with each trader interface is a default context. The default context determines the maximum visibility through a trader interface.

### 3.3 Export contexts

Contexts are used by an exporter to selectively expose a service offer to potential importers.

All exporters of a trader at an interface have access to the same set of export contexts. In a unfederated trader, the export contexts are just the trader contexts of this trader. Section 3.7 discusses export contexts in a trader federation.

### 3.4 Import contexts

Contexts are used by an importer to limit the visibility of service offers to only those of interest to the importer.

When importing, an importer can nominate one or more import contexts. Any service offer returned by the import operation must satisfy the membership rules of each of the nominated import contexts. However, not all service offers satisfying the nominated import contexts will be visible to the importer. Service offers returned to the importer might be further restricted by enterprise rules (e.g. security) imposed by the exporter or trader administration.

The import contexts available to an importer (of an unfederated trader) include all of the trader contexts available at the importer's interface. In addition, import contexts can be defined as

import context views; these are associated with either an import interface (and shared by all importers using that interface) or an importer alone.

Figure1 illustrates the relationship of trader contexts, export contexts, and import contexts. The trader contains three trader contexts with membership of upper-case letters, lower-case letters, and numbers. The exporters S1 and S2 are associated with the interface through which only the upper-case and number contexts are accessible, while exporters T1 and T2 can access only the lower-case and number contexts. The import contexts of importer P1 are the same as the trader contexts (the default) while importer Q1 has constructed the import context views of even numbers and odd numbers.

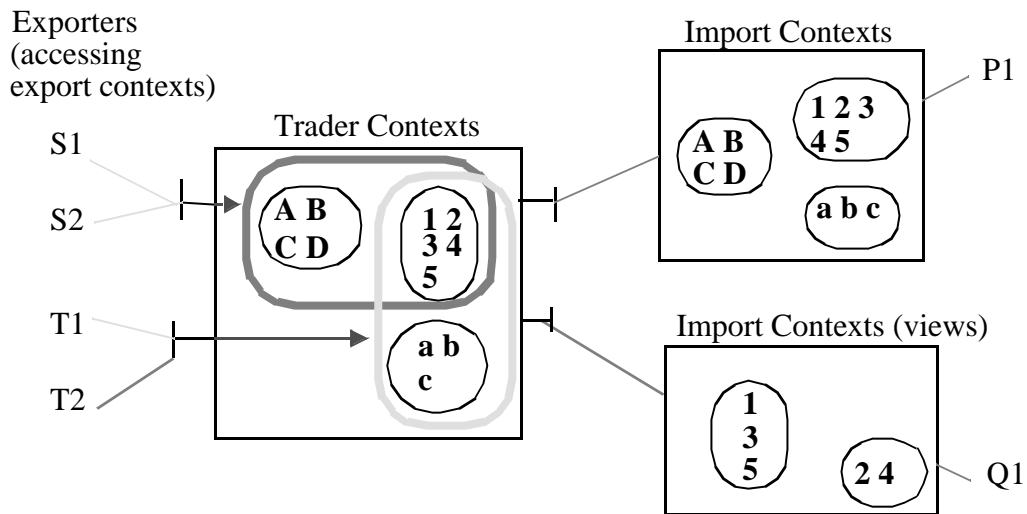


Figure 1. Export Contexts, Trader Contexts, and Import Contexts

### 3.5 Trading context

When traders federate, the set of offers available for import are expanded to include service offers from individual trader contexts that each trader is willing to share. This total set of service offers available through the federation of traders form a trading context.

In an unfederated trader, an import context (including import context views) must be a subset of the trader context, i.e. all of the service offers within the import context must be held by the trader. Once federated, an import context can include service offers available through the federation. Hence, import contexts are subsets of the trading context.

Figure2 shows the trader of Figure1 as part of a trader federation. The other trader, Trader B, groups service offers by their position within their ‘‘alphabet’’ (i.e. ‘X’ is the 24th letter of the alphabet). After federation, the importer contexts include service offers from both traders.

Note that it is the trader enterprise policy that determines the circumstances and extent to which a local trader will search through its federated partners. For example, the federated traders might only be queried when an import operation cannot be satisfied locally. Other traders might allow the importers to specify the extent of the search, either explicitly or implicitly in terms of cost or time.

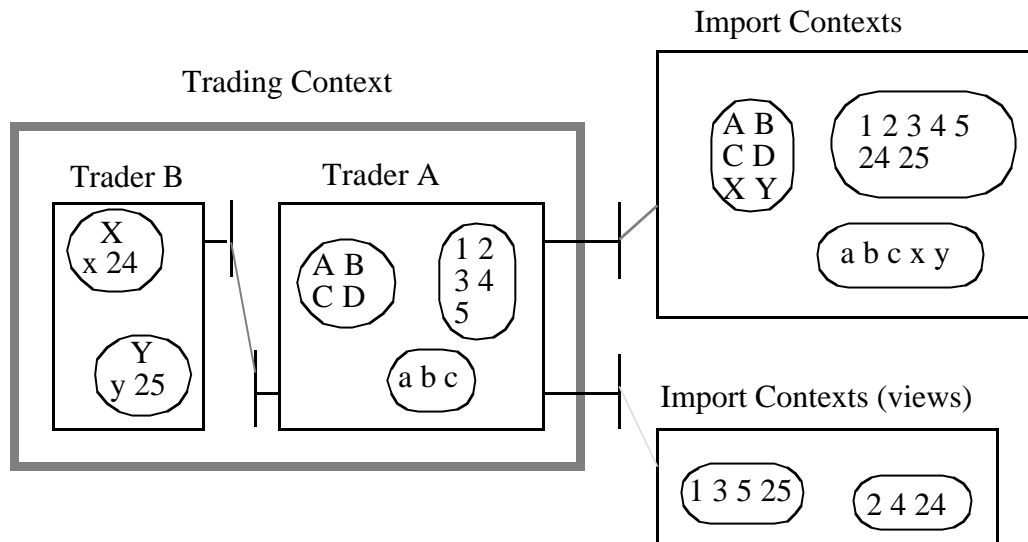


Figure 2. Trading Context and Import Contexts

### 3.6 Context management in Federation

Since contexts have membership rules, an import context equates to a clause in the matching constraint. That is, a local context can be expanded into its associated rules and the rules be used as additional matching constraints in a request to a remote trader. Such rule expansions enables the support of a local user's familiar context view in a federation of traders.

Since rule-based contexts do not require context sharing in a federation, each trader can provide any trader context structure it desires for its local users and each importer can maintain its own local context view. The local context structure can continue to be used during federation. The only effect on users is that context population will (probably) increase upon federation (as shown in Figure2).

Since local context and matching constraints will be expressed in terms familiar to a local user, federation must be based on criteria that are accepted and known to all traders. The only criteria that are accepted and known by all traders is the type hierarchy. Therefore, for federation, each trader's search space can be regarded (but not necessarily implemented) as being partitioned by type hierarchy. This is appropriate as open distributed processing requires strong typing. Thus, the type hierarchy can be used as the conformant structure for federation and all trader-trader interactions will be based on service types and service properties (again, strongly typed). Therefore, a trader must expand its local contexts into membership rules (perhaps recursively) until these rules are expressed solely in terms of service types and service properties.

### 3.7 Exporting to Federated Traders

Federation offers exporters the potential to advertise their services more widely, either by:

- their service offer (advertised within their local trader) being accessible via federated import operations, or
- advertising the service offer within a remote trader.

An exporter can advertise a service offer within a remote trader by either:

- becoming a direct user of the remote trader (the only method possible in the absence of federation), or
- exporting into an export context offered by the local trader but *belonging to a remote trader*, as described below.

Although a trader does not need to learn anything of the context structure of another trader, it is not prevented from doing so. If a remote trader contains trader contexts that would further the goals of the local trader's community, then the local trader can offer that remote trader context as a local export context. Any exports placed into that export context would not be held by the local trader but would be "re-exported" to the remote trader.

So, while an export context is a trader context, it is not necessarily a *local* trader context.

Note that a local trader could also create its own trader context with the same membership rule as a remote trader context, if desired. Exports to that context would be maintained by the local trader.

#### 4 COMPUTATIONAL VIEW

Computationally, a trader:

- stores exported service offers,
- accepts definitions for contexts for export and import, and
- matches service offers, constrained by export contexts and trader context, with service requests, constrained by import contexts and trading context.

The trader offers trading interfaces and context management interfaces. For a trading interface, the operations are:

- *export* - to store service offers within the export context(s) associated with the trading interface of the exporter.
- *import* - to match service requests with service offers within the view of the requesting importer's import context

For the context management interface, operations are required to allow users to define context membership rules and manipulate relationships between contexts. Other operations provide information about context identifiers, context rules and context structures. Thus, the operations are:

- *create\_import\_context* -to create membership rules for a context which is specific to an importer and associate the rules with an identifier or name. The operation can (optionally) link the new context with existing contexts via context relationships.
- *create\_export\_context* - to create an export context with a specific set of rules and associate the rules with an identifier. Again, the new context can be related to existing contexts.

- *delete\_context* - to dissociate an identifier with a set of context rules (for either exporter or importer). The identifier can no longer be used as a short-hand notation for the context rule. All links (via relationships) to this context are removed.
- *list\_context* - to list the details of a context. It lists the context identifier, the context membership rule, and relationships to other contexts. If the membership rule consists of other context identifiers, the operation will be applied recursively (if requested).
- *list\_content* - to list the service offers stored in a context. Since the purpose of the operation is to discover what is within a context, a user can request the context's content be described as total number of service offers or service offers (with all service properties, common service properties only, or no service properties).

## 5 ENGINEERING VIEW

The search for matches of importer's requests with service offers can be optimised if knowledge of the relationship between trader contexts and import contexts are known. An import request nominates a context (expressed as rules) and a set of requirements on service properties. Let  $X$  = context on import and  $A$  = trader context, then:

$X \equiv A$	search only A checking only other properties
$X \Leftrightarrow A$	search only A checking only other properties
$X \subseteq A$	search only A but checking all (other properties + membership rules)
$X \supseteq A$	search A checking only other properties search $\sim A$ , checking all
$A \cap X \neq \emptyset$	search all service offers, checking all
$A \cap X = \emptyset$	search $\sim A$ , checking all

If there is no knowledge of the relationship between trader contexts and import context, then no optimization is possible; all service offers must be tested for both membership rules and constraints on other properties.

Thus, for optimal search efficiency, mappings need to be maintained:

- between context identifiers and context predicates
- between import contexts and trader contexts
- between trader context and service offers

It will be a trader's engineering choice to implement import contexts by either:

- computing the membership of the import context in response to an importer's request
- maintaining the mapping between an import context and each service offer which qualifies for membership. When a new export is received by the trader, these mappings will need to be updated. This approach is viable only if the number of imports far exceeds the number of exports.

Optimization is also possible for federated traders. Federation can be optimized if a trader understands the contexts of another trader. One trader can inspect and adopt all, or any, of the contexts of a remote trader if it so desires. This is easily done by learning the remote trader's context with the *list\_context* operation and incorporating that context into its own context structure with the *add\_context* operation.

## 6 CONCLUSIONS

In this paper, we have proposed that a membership rule is associated with a context. The context membership rule can be used to express requirements of the exporters, the importers, and the trader administration. The context identifier is merely a short-hand notation for these requirements.

The use of rule-based contexts was motivated by the need for a method of trader federation that is transparent to the users, without affecting the heterogeneity and autonomy of the traders within the federation. Rule-based contexts allow users to view the trader federation through the locally-defined context structures. Traders can federate without knowing the context structures of the other traders. However, if a trader is prepared to learn the context structure of another trader, engineering optimisations can be employed.

Rule-based contexts allow the definition of context views, which can provide importers with an individualized context structure reflecting their requirements. These context views are transparently mapped into matching constraints applied to the search of the trader database.

## Acknowledgements

The work reported in this paper has been funded in part by the Cooperative Research Centres Program through the department of the Prime Minister and the Cabinet of the Commonwealth Government of Australia. This research was also supported by Telecom (Australia) Research Laboratories through the Centre of Expertise in Distributed Information Systems (CEDIS).

## References

- [1] ISO/IEC JTC1/SC21/WG7/N743, Working Document on Topic 9.1 - ODP Trader, November 1992.
- [2] Architecture Projects Management Ltd (APM), "ANSAware 3.0 Implementation Manual", January 1991.
- [3] Barr W., Boyd T., and Inoue Y., "The TINA Initiative", IEEE Communications Magazine, March 1993.
- [4] CCITT Recommendation X.501 (ISO/IEC JTC1/SC21 ISO 9594-2), "The Directory - Models", March 1988.