Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

Training

ANSAwise - Using the ANSAware Trader service

Chris Mayers

Abstract

Organizations wish to deploy large-scale distributed systems.

Trading is a concept fundamental to distributed systems services.

This module of the ANSAware training programme reviews the role of trading in general and describes the ANSAware Trader service. Participants then use the ANSAware trading tools (both GUI and command-line) to experiment with their effects. Finally, participants design the use of Trader properties and modify the Simple Bank client and server to use these properties.

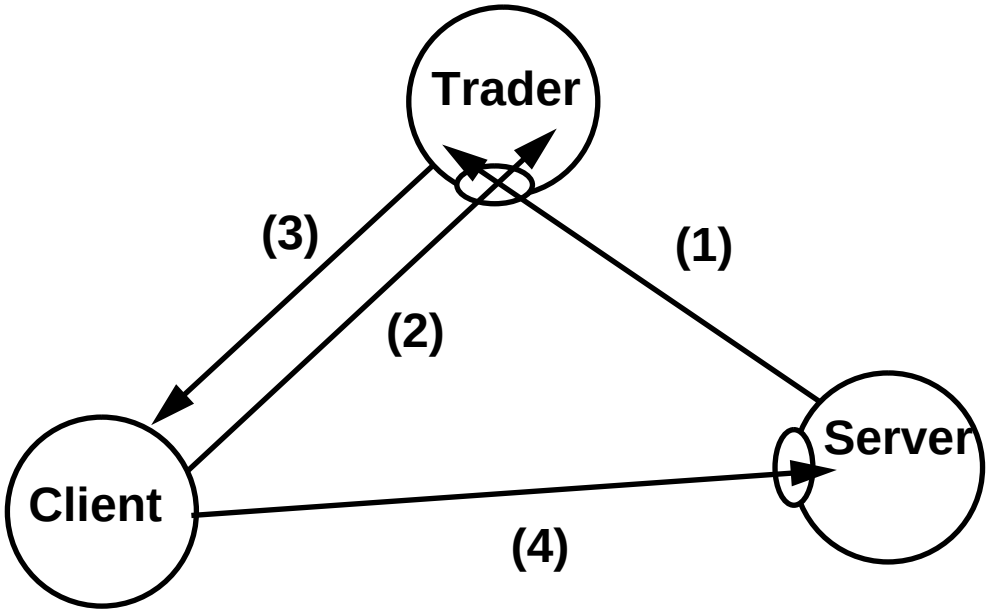
APM.1585.01 **Approved** 2nd October 1995

Briefing Note

Distribution:
Supersedes:
Superseded by:

ANSAware 4.1

Using the Trader service



In this session

- **Explain how the ANSAware Trader implements the ANSA trading concepts**
- **Describe the command-line tools to manipulate the Trader**
- **Experiment with using these tools**

Review of ANSA trading concepts

- Trading allows clients to find servers that provide the services that they need
- The main criterion for matching is type conformance
- Traders can be federated together

Purpose of the ANSAware Trader service

- To implement most of the ANSA trading concepts
- To allow a client to find a server which offers a service:
  - of an appropriate (matching) type
  - with other appropriate characteristics (for example, Quality of Service)
- To support federation of multiple traders
- To allow administrative management of service offers

Information in the Trader

- **The ANSAware Trader holds information about:**
  - **service offers and their properties**
  - **interface types**
  - **trader contexts**

The ANSAware Trader interfaces

- The Trader service has 4 separate interfaces:
- 1 service interface (of type Trader)
  - for registering, looking up, and deleting service offers
- 3 management interfaces (of types TrType, TrCtxt, TrFed)
  - for managing interface types
  - for managing trader contexts
  - for managing trader federation

The Trader's service interface *(Trader)*

- **It has 3 operations: Register, Lookup, and Delete**
  - ***Register*: exports a new service offer to the trader**
  - ***Lookup* - lists one, or all, offers that match**

    e.g. `traderRef$Lookup(...)`
  - ***Delete*: removes service offers**

Using the Trader service interface

- **Applications don't generally use these Trader operations directly**
  - **it is simpler to use the $Export, $Import, and $Withdraw PREPC statements instead**
  - **these PREPC statements invoke the Trader operations**
  - **traderRef$Import invokes Lookup**
  - **traderRef$Export invokes Register**

Trader service programs

- The Trader is supplied with some simple command-line programs
- The program `trtest` invokes 1000 Lookup operations and prints how long it took
- The program `trclient` invokes a Search

```
trclient search ansa /
```

Compatibility of interface types

- If an interface-type StringOps is compatible with type Echo, it provides at least the operations of type Echo
  - ... it may provide extra operations as well
- A new interface that is an extension of an existing one can be defined in IDL with the IS COMPATIBLE WITH statement:

```
StringOps: INTERFACE =
IS COMPATIBLE WITH Echo;
BEGIN
-- Define new operations here...
END.
```

- This is called type conformance in the Computational Model

Effect of type compatibility

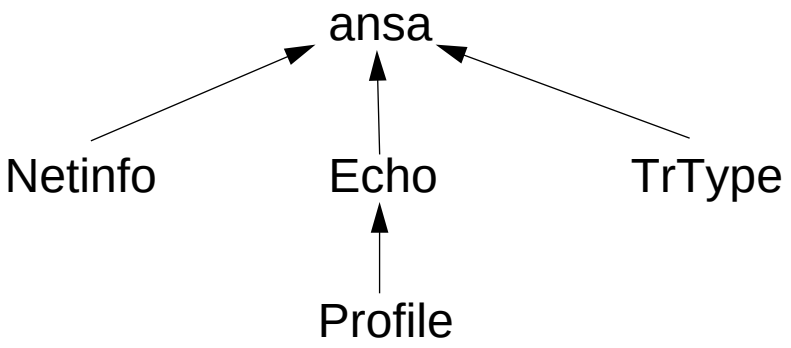
- **Because StringOps conforms to Echo, it can be used wherever Echo is needed**
  - **a client that imports type Echo may obtain an interface for either an Echo or a StringOps service**
- **code for StringOps must provide all the operations of Echo**
  - **and it can have others**
- **StringOps type must also be registered with the trader**

Trader interface types

- Every interface type has a name
  - as given in the IDL
- Each type has one or more immediate supertypes, to which it conforms
  - The root type is conventionally called ansa (it has no supertype)
  - Relationship of new type to others is given when registering type with Trader
  - Trader maintains a list of the type relationships

The Trader type graph

- **The relationships form a directed acyclic graph**
  - **more general than a tree, because types can have more than one supertype**



- **Conformance of offer type to requested type ensures that client and server can exchange information**
  - **Type relationships are asserted by user, and checked at invocation time**
  - **Obviously, nothing can check whether the interface is correctly implemented; this depends on the implementations of client and server**

TrType - managing the type-space

- **Trader's TrType interface provides the operations for managing Trader's type-space**

- **The command-line tool `typecl` uses this interface to manipulate trader's type-graph**

Managing types with typecl

- With the `typecl` (type client) program you can:
  - add a new type (optionally specifying types to which it conforms)
  - delete an existing type
  - mask an existing type (that is, prevent further use before deleting it)
  - unmask a masked type
  - list existing types

Adding types with typecl

- To add a new type:

```
typecl add type_name [supertype [supertype...]]
```

  - for example:

```
typecl add Newtype ansa
```

- The supertype must already exist...

```
typecl add NewType ansa/SBank
add failed - status = TNoSuchSuperType
```

  - .. and the new type must not already exist

```
typecl add SBank ansa
add failed - status = TAlreadyExists
```

- You can specify more than one supertype (in this case, NewType conforms to t1 and t2)

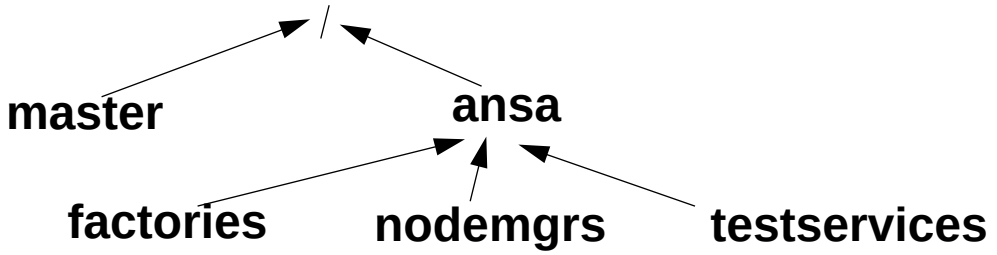
```
typecl add NewType t1 t2
```

Trader contexts

- Contexts allow administrators to group service offers together
- Every offer is registered in a context
- Trader maintains a tree of contexts
  - each context can register many service offers
  - contexts can contain other contexts within them (like Unix subdirectories)
- Trader uses context pathnames (like Unix pathnames)
  - the root of the context tree is "/"

Administering Trader contexts

- **The context tree looks like this:**



- **With the context client (`ctxtcl`) program you can add, delete, and list contexts**

Adding Trader contexts with ctxtcl

- **To add a new context:**

```
ctxtcl add context
```

  - **... for example:**

```
ctxtcl add /ansa/testservices/course
```

Deleting Trader contexts with ctxtcl

- **To delete a context:**

```
ctxtcl delete context
```

  - **... for example:**

```
ctxtcl delete /ansa/testservices/course
```


Listing Trader contexts with ctxtcl

- To list contexts:

```
ctxtcl list
```

  - for example:

```
ctxtcl list

/master
/ansa
/ansa/factories
/ansa/nodemgrs
/ansa/testservices
```


Trader properties and constraints

- **Properties hold application-specific information about a service offer**
  - **e.g. for a print service: LinesPerMinute, PaperSize, costPerPage, Turnaround**
- **Property Name/value pairs are specified when offer is registered**
- **The Trader automatically generates some properties**
  - **for example, it generates the (Type, TypeName) property for every offer registered**
- **Clients requesting services may specify:**
  - **Acceptable property constraints (e.g. PaperSize==’A4’)**
  - **Selection of offer with maximum (or minimum) value of a property**

PREPC Export and Import syntax

- **Export and Import PREPC statements have different syntax for specifying properties:**

  - `Export( type, context, "Node machine", ifref)`

  - `Import( type, context, "Node==’machine’")`
    specify constraint-expression (more on this later)

  - `Export( type, context, "Node machine Name Jane ...", ifref)`
    space-separated list of name-value pairs

Trader properties when Exporting

- When exporting a service you specify a property list:
  - the list consists of pairs of (name, value) items
  - the pairs in the list separated by spaces
  - for example:

```
Node basilisk Price 100
```


Trader properties when Importing

- When importing a service you specify constraints:
  - the constraints are in the form of a logical expression, for example:

```
(Node == 'basilisk') and (Price <= 200)
```

  - the expressions can use the usual operators (`==`, `!=`, `and`, `or`, `not`, ...)

- For the full syntax of the constraint language, see *Application Programming in ANSAware* (section 3.11.4)

Finding one matching offer using trclient

- To look up one matching offer:

```
trclient lookup type context [constraints]
```

  - for example:

```
trclient lookup ansa /
```

- This displays one offer, selected randomly from those which match

Looking up and matching one offer randomly

- **Matching is determined by the type, context, and any property constraints**
- **Selecting the offer randomly spreads the client load across servers with the same interface**
- **This is how the PREPC $Import statement works**

Finding all matching offers using trclient

- **To search for all matching offers:**

```
trclient search type context [constraints]
```

  - **for example:**

```
trclient search ansa /
```

- **Matching for type, context and constraints is the same as Lookup**

  - **but you see all the matching offers, not just one of them**

Registering an offer using trclient

- **To register an offer:**

```
trclient register type context [constraints]
```

  - **for example:**

```
trclient register test3 /ansa “Name Foo” 60
```

Deleting an offer using trclient

- To delete an offer:

  ```
  trclient delete nonce [constraints]
  ```

- A nonce is a unique identifier for the offer generated internally by ANSAware
  - it is one of the properties displayed by `trclient search`
- You shouldn't normally need to delete an offer
  - Services normally withdraw their offers when terminating...
  - ...you only need to delete the offer using trclient if the service terminates incorrectly

Local and Master Traders

- **Every ANSAware program knows the interface-references of two Traders**
  - **the Local and Master Traders**
  - **the interface-references are compiled in to the capsule library**
  - **the Local and Master Traders may in fact be the same Trader**
- **When importing, the Local Trader is tried first: if there is no match, the Master Trader is tried**
- **This is not true federation; it is an additional mechanism for grouping services**

X Graphical User Interface to the Trader

- **These are equivalent to the command-line tools:**

Interface	Command-line tool	X tool
Trader Client	trclient	offerMgr
Trader Type Client	typecl	typeMgr
Trader Context Client	ctxtcl	ctxtMgr

- **The X tools are not built and installed by default**

Exercises

- Experiment with the Trader tools
- Identify your bank service
  - Export with a specific property value, for example:

    `"BankName ‘MyBank PLC’"`
- Change the teller and manager client programs to use only your bank service
  - Import with a corresponding property constraint, for example:

    `"BankName == ‘MyBank PLC’"`
- Verify that both client and server programs are working as expected
- Consider what other properties might be useful for the bank service

Your notes

ANSAware Trader - Summary

- **Trader holds information about:**
  - **service offers and their properties**
  - **interface types**
  - **trader contexts**
- **Traders can be federated**

ANSAware Trader - More information

- **For the command-line tools**
  - see Appendix C of *Application Programming in ANSAware*
- **For the X tools**
  - also see Appendix C of *Application Programming in ANSAware*
- **For an overview of the ANSAware Trader**
  - see Chapter 3 of *Application Programming in ANSAware*
- **For the Trader's interface definitions**
  - see Appendix D of *Application Programming in ANSAware*

ANSAware 4.1 Trader - Extra information on Types

- The ANSA computational model is not fully implemented by the Trader

- Types:

  - conformance relationships must be specified explicitly when types are added. If the relationship is incorrect, the Trader will not detect it. But there is a sanity check at invocation time

  - the Trader does not check conformance of types. It merely compares names

  - programs assert the types of their interfaces; these assertions may be invalid. The Trader cannot check this either, since it does not have the type description available

ANSAWare 4.1 Trader - Extra information

- **Contexts**
  - the name space is a tree rather than a directed acyclic graph. Context name components are separated by a "/".
- **Properties**
  - All properties are optional. There is no way of forcing an offer of type X to specify properties (Name, x), (Host, y).
- **Searching**
  - Searching is ‘shallow’; federated traders are only consulted if the current trader cannot satisfy the import request.

ANSAWare 4.1 Trader - Deleting service offers

- **Service offers are deleted when:**
  - **a server terminates normally: the server can explicitly withdraw any offers it has `Exported`, using the PREPC `Withdraw` instruction**
  - **a server is killed by a signal (CTRL-C): the ANSAware infrastructure withdraws all `Exported` offers from Trader before exiting the capsule**
  - **an offer is 'stale': the trader will automatically remove an offer it considers 'stale'**
  - **a client cannot invoke operations on an interface-reference received from the trader: the client's infrastructure notifies the Trader**