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 - Simple Bank Exercise Guide

Chris Mayers

Abstract

Organizations wish to obtain practical experience in designing distributed applications. Within a training course, a hands-on exercise is the most effective way to provide that experience.

CNET have requested such an exercise for inclusion within their training programme for distributed systems.

This document describes the Simple Bank Exercise, which uses ANSAware 4.1. This document is intended for the course presenter, and describes the aims and objectives of the exercise, how the exercise is structured, and the points that are likely to arise from it.

(The components of this exercise may also be used in different ways in different training courses.)

APM.1547.02 **Approved** 2nd October 1995

Briefing Note

Distribution:
Supersedes:
Superseded by:

ANSAwise - Simple Bank Exercise Guide

ANSAwise - Simple Bank Exercise Guide

Chris Mayers

APM.1547.02

2nd October 1995

The material in this Report has been developed as part of the ANSA Architecture for Open Distributed Systems. ANSA is a collaborative initiative, managed by Architecture Projects Management Limited on behalf of the companies sponsoring the ANSA Workprogramme.

The ANSA initiative is open to all companies and organisations. Further information on the ANSA Workprogramme, the material in this report, and on other reports can be obtained from the address below.

The authors acknowledge the help and assistance of their colleagues, in sponsoring companies and the ANSA team in Cambridge in the preparation of this report.

Architecture Projects Management Limited

Poseidon House
Castle Park
CAMBRIDGE
CB3 0RD
United Kingdom

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

1 Introduction

1.1 Audience

This document is intended for course presenters of the Simple Bank Exercise.

1.2 Scope

This version of the document is specifically intended for the CNET project [Proposal].

1.3 Status

This is a draft for comment.

2 Overview

2.1 About the Simple Bank Exercise

This exercise is based on the standard Simple Bank sample application supplied with ANSAware [AWAP4.1]. It is developed from a similar exercise used in the ANSAware-specific [Course].

2.2 Aims of the Simple Bank Exercise

The aims of the Simple Bank Exercise are to demonstrate:

- that developing distributed applications is straightforward
- what a specification can and must contain
- the challenges of non-transparency for distributed applications

This exercise does not aim to teach the use of ANSAware itself; ANSAware is a means to an end. ANSAware 4.1.1 is used in this exercise because it provides all the features necessary to demonstrate distributed systems concepts, and because it is simple to learn.

2.3 Objectives of the Simple Bank Exercise

On completing the Simple Bank Exercise, participants will be able to:

- Specify an interface in ANSAware IDL
- Build simple ANSAware applications (client and server)
- Implement a client application using ANSAware PREPC
- Implement the server
- Add error-handling to a client application
- Add concurrency control to a server
- Use the ANSAware Trader and Factory services

This exercise is not intended for self-study by participants.

The following topics are not within the scope of this exercise:

- installing and configuring ANSAware
- porting ANSAware applications (between platforms)
- advanced testing and debugging
- replication
- use of the ANSAware real-time features (in ANSAware/RT)
- security
- persistent storage

2.4 Participant prerequisites

Participants in the Simple Bank Exercise must:

- have attended the course *Introduction to Design of Distributed Systems*
- have basic UNIX user skills (basic commands and use of a Unix text editor)
- be fluent in C
- have 2-3 years industrial software engineering experience

No previous experience with ANSAware is required. In fact, because this is a basic exercise, participants who have already had previous experience with ANSAware may not need to attend this exercise.

3 Structure of the Simple Bank Exercise

3.1 Overall structure

The Simple Bank Exercise takes 1 day of class time. It is intended to be delivered immediately following an Introduction to Design of Distributed Applications course; therefore only a minimal review of basic principles is required.

The exercise starts with a standard 10-minute Welcome session, and finishes with a standard 10-minute Roundup. The exercise proper is structured as five modules each lasting 1.5 hours. These modules are intended to be presented together, in the order specified below; they are not stand-alone, and cannot be used independently.

Normally two modules would be delivered in the morning, one module would span the lunch breaks, and two further modules would be delivered in the afternoon.

Each module consists of a standard training presentation (approximately 30 slides). Each presentation is in two halves; an introduction explaining what the participants will do, and a summary (debrief) explaining what the participants have learned. Each half of the presentation will last about 10 minutes, including discussion and questions/answers. Participants therefore have more than 1 hour to complete the activity.

Each module consists of a single activity. Each module starts with a new set of source code. Thus, whether participants succeed or fail to complete one activity, they all start on an equal basis on the next activity.

Participants work in pairs.

The modules are:

- Welcome [Welcome]
- Building Applications with ANSAware Tools [Building ANSAware Applications]
- Using the ANSAware Trader service [ANSAware Trader]
- Exploiting Concurrency in ANSAware Applications [ANSAware Concurrency]
- Implementing Robust ANSAware Applications [Robust ANSAware]
- Managing ANSAware Applications [Managing ANSAware Applications]
- Course Roundup [Roundup]

3.2 Welcome

This brief session welcomes the participants and explains the purpose and structure of the course.

3.3 Building Applications with ANSAware Tools

This session:

- explains why ANSAware has been selected for this exercise
- gives some examples of systems which use ANSAware
- explains briefly how ANSAware compares with CORBA
- describes ANSAware IDL
- shows the steps to build an ANSAware client and server
- shows the overall structure of a simple ANSAware client and server
- shows how to run an ANSAware client and server

This session then explains the overall setting for the Simple Bank exercise, and why it has been chosen.

The participants are supplied with incomplete source code for Simple Bank. The Access operation of the Account interface does not contain the logic for validating the PIN against the account number. An outline is provided, with comments indicating how the operation should be implemented.

Participants then:

- complete the specification of the Banking/SBank interface in ANSAware IDL
- complete the implementation of a client of the Simple Bank service, in ANSAware PREPC, and C
- complete the implementation of the server in ANSAware PREPC, and C
- test the functioning of client and server

(Note that although the source code is incomplete, participants are supplied with complete makefiles, so building the client and server applications is straightforward.)

Note that the ANSAware Trader has not yet been described. Therefore the existence of the Trader must be transparent to the participants. Furthermore, the clients and servers are being modified by different participants; each participant's client must use their own server. This will require some specific configuration for each participant. (The following module will explain the functioning of the Trader.)

3.4 Using the ANSAware Trader service

In this session:

- the role of trading in ODP is reviewed
- the importance of type compatibility is described
- the ANSAware Trading service is described

Participants use the ANSAware Trader tools (both GUI and command-line) to:

- inspect service offers
- configure interface types in the Trader
- configure Trader contexts
- delete stale service offers (to recover from errors)

In this session, participants also:

- design the use of Trader properties by the Simple Bank service
- modify the Simple Bank client and server to use these properties

3.5 Exploiting Concurrency in ANSAware Applications

This session explains:

- how concurrency can improve application performance
- transactions and the ACID properties
- simple concurrency control using locks
- implications of using concurrency features in C
- how to use ANSAware basic concurrency features (mutex locks)

Participants then:

- modify the Simple Bank server application to support multiple concurrent clients
- configure the Simple Bank server application to support multiple concurrent clients
- test the concurrent behaviour of the modified Simple Bank server

This session briefly mentions, does not cover in detail:

- the explicit use of ANSAware multi-threading (fork/join)
- asynchronous RPC (initiate/redeem)
- optimistic concurrency control techniques

Note that the ANSAware concurrency model is more sophisticated than that used by POSIX/DCE Threads (pthreads), and the traditional approach taken in database concurrency. However the basic concurrency features are the same, and this session concentrates on them.

3.6 Implementing Robust ANSAware Applications

This session explains:

- why distributed applications must be able to handle a wider range of error conditions than centralized applications
- how to use ANSAware exceptions to handle these error conditions

Participants then modify the client application to:

- use ANSAware exception handling to report application-specific error messages for errors detected by the infrastructure (for example, when a server is not running)
- override the default relocation policy (for retries)
- report application-level error messages (for example, security violations)

Participants also test the behaviour of the modified client application when these error conditions occur.

The ANSAware exception handling mechanisms are not as powerful as ANSA terminations (or, indeed, CORBA exceptions). Furthermore, the language

mappings for exceptions are very different (ANSAware does not use C exception macros). This session concentrates on the principles of exception handling.

Note that this session does not cover memory management issues. Although this is an important aspect of implementing robust distributed applications, it is not within the scope of this exercise.

3.7 Managing ANSAware Applications

This session explains

- the different methods of creating and activating a server
- the role of the Factory service
- the role of the Node Manager service, and its integration with the Trader
- managing the services with ANSAware

Participants then modify the Simple Bank server to:

- be manageable via the Factory and Node Manager
- handle new error conditions accordingly

Note that the Simple Bank source code shipped with ANSAware 4.1.1 does not support being managed; it is in fact suggested as an extension. So in this session, participants are adding real functionality to a standard example.

3.8 Course Roundup

This is a 10-minute session. It:

- summarizes what has been learned during the exercise
- lists some additional features of ANSAware that have not been covered by the exercise
- explains the evolution of ANSAware, in ANSAware/RT, and also towards CORBA
- explains how to find out more about ANSAware, and distributed systems products in general

Some of these topics will be covered in more detail in the Advanced Design of Distributed Systems course.

4 Course presenter information

4.1 Bill of materials

Each participant should be supplied with a copy of the slides for each session.

The slides are intended for printing two-rows per-page (2 up). They can be copied double-sided.

4.2 Equipment requirements

All workstations should be Sun Sparcstations (any model of Sparcstation will run ANSAware 4.1.1). The Sparcstations must run SunOS 4.1.1[1]. (The exercise will not work correctly with Sun's Solaris 2.x.)

Calculate the number of Sparcstations as follows:

- 1 for each two course participants,
- plus 1 for the course presenter
- plus 1 spare

It is recommended that a separate Ethernet segment be used, to avoid network disruption to other users.

Participants will require access to the usual tools:

- text editor
- C compiler (supported by ANSAware 4.1.1)

It is recommended that participants bring along copies of their Unix user configuration files if they wish.

4.3 Presenter prerequisites

The course presenter must:

- have presented the *Introduction to Design of Distributed Systems* course
- be experienced with building ANSAware applications, and know how to recover from common errors
- be familiar with known problems with ANSAware 4.1.1, including those described in §4.4 *ANSAware installation and maintenance*
- complete the Simple Bank exercise themselves

1. It is a coincidence that the same version numbers of SunOS and ANSAware are required. They are entirely unrelated.

4.4 ANSAware installation and maintenance

4.4.1 Installing the Simple Bank exercise source code

If ANSAware has been installed for the same machine, no changes are needed to the source code supplied for the exercise.

If ANSAware has been installed for a different machine, it is necessary to edit the Simple Bank `makefile` as follows:

- change the definition location of the installed ANSAware files. for example:

```
TOP= /usr/groups/ANSAwareDev/install/sun4_sos_4.1
```

- remove all the makefile dependency lines which start with the text `Makefile::` Other makefile dependencies should left as they are

```
Makefile::
```

4.4.2 ANSAware Trader stale offers

If a server crashes, it will not always withdraw its service offers from the Trader. This is particularly likely when course participants are debugging their applications. A service offer that has been left in the Trader when the server is no longer running is known as a stale offer. (Do not confuse a stale offer with a proxy offer; a proxy offer is used by the Node Manager to start up a service on demand.)

It is intended that ANSAware should automatically detect and remove stale offers under most circumstances. However, in practice, ANSAware often does not do so. The likely symptoms include:

- client applications which run correctly on some occasions, but not on other occasions
- clients failing to bind to servers

To search for stale offers, type:

```
trclient search ansa /
```

Check this frequently during each session.

The ANSAware Trader `trclient purge` operation does not always remove stale offers. It is therefore necessary to use the `trclient delete` option. This requires typing in an ANSAware nonce, which is a long digit string. It is therefore strongly recommended that the course presenter use a terminal that is set up to support cut-and-paste easily.

Remember that the Simple Bank has several interfaces. Each interface has separate offers, and must be purged separately.

It is also possible for the Trader's own offers to become stale, if the Trader crashes. Remember that the Trader also has several interfaces, and includes the Relocator service.

To distinguish active and stale offers for the Trader's own interfaces, check the capsule id (cid) for each offer. The capsule id is the Unix process id (pid). If the cid shown by `trclient` does not match the pid shown by `ps`, the offer is stale. These offers can be removed by `trclient delete` in the usual way.

4.4.3 Standard server startup and shutdown order

Start up servers in the order:

1. Trader
2. Factory
3. Node Manager

Shut down these servers in the reverse order. Starting up servers in the wrong order can result in stale offers.

4.4.4 Multiple servers

Beware of starting more than copy of a standard server (Trader, Factory, or Node Manager.) Use the Unix ps command to detect this.

4.5 Advice for particular sessions

4.5.1 Welcome

Emphasise that ANSAware is being used as a vehicle for demonstrating distributed systems principles.

4.5.2 Building Applications with ANSAware Tools

Emphasise that error handling has been deliberately removed from the source code, and that participants will have the challenge of adding back the error handling later (in §4.5.5 *Implementing Robust ANSAware Applications*). Meanwhile, participants should not be surprised if Simple Bank crashes when errors occur.

Be prepared to help course participants find their way around the source code.

4.5.3 Using the ANSAware Trader service

This session should be straightforward for participants. However, take particular care to watch for stale offers in this session. It is also possible that participants may inadvertently use each others Traders, so watch for confusion here.

The X Windows Trader tools are not required for this exercise. The course presenter may wish to demonstrate them, however.

4.5.4 Exploiting Concurrency in ANSAware Applications

This session should be straightforward for participants. The main difficulty is in testing the locks are operating correctly. There is no easy way to convince yourself of this.

Note that participants only add locking to one interface. A full solution would require locking for all the other interfaces as well. This would not add any further interest to the exercise, however.

In this exercise, there are comments in the source code explaining where the locks should be placed. In a real application, determining this would be a large part of the design challenge.

4.5.5 Implementing Robust ANSAware Applications

This is the hardest session. This is simply because error handling in distributed application is inherently difficult. It may be an unfamiliar challenge, but it is a challenge that the application programmers must overcome.

Expect to provide support and guidance throughout this exercise.

Depending on the participants, this exercise may take longer than 1.5 hours.

4.5.6 Managing ANSAware Applications

Depending on the progress made by participants, there may not be time to start this exercise, in which case it can safely be omitted.

The exercise is in two halves; the Factory, and the Node Manager. In the Factory part, the challenge is to modify the code in the server. In the Node Manager part, the challenge is in using the Node Manager tools to configure the service correctly.

The Factory and Node Manager tools are rather complicated to use. Familiarize yourself with all their options in advance.

4.5.7 Course Roundup

The slides in this session are a very brief summary. The course presenter should elaborate on the actual difficulties that the participants met, and how they overcame them.

It is important to present the positive side of the day's learning!

References

[AWAP4.1]

ANSAware 4.1 - Application Programming in ANSAware
RM.102.02

[AWSM4.1]

ANSAware 4.1 - System Manager's Guide
RM.100.02

[AWSP4.1]

ANSAware 4.1 - System Programming in ANSAware
RM.101.02

[Proposal]

Proposal for Training in Distributed Systems [CNET]
APM.1450

[Course]

Writing Distributed Applications using ANSA and ANSAware
APM.1261

[Exercises]

ANSAwise - case studies and exercises
APM.1401

[Welcome]

ANSAwise - Welcome, to Hands-on Design with ANSAware
APM.1589

[Building ANSAware Applications]

ANSAwise - Building Applications with ANSAware Tools
APM.1584

[ANSAware Trader]

ANSAwise - Using the ANSAware Trader service
APM.1585

[ANSAware Concurrency]

ANSAwise - Exploiting Concurrency in ANSAware Applications
APM.1586

[Robust ANSAware]

ANSAwise - Implementing Robust ANSAware Applications
APM.1587

[Managing ANSAware Applications]

ANSAwise - Managing ANSAware Applications
APM.1588

[Roundup]

ANSAwise - Course Roundup, to Hands-on Design with ANSAware
APM.1590