SWISS EXCHANGE SWX
ERM Data Distribution Centre Sub-System Specification

I-ERM-DDC-100B/E, Draft Version 1.0B, 17 Sep 98

Identification

Title:	ERM Data Distribution Centre Sub-System	
Version:	Draft Version 1.0B, 17 Sep 98	
Classification:	For Internal Use Only	
Intended Audience:	ERM Fachleiter	
Keywords:		
Reference:	I-ERM-DDC-100B/E	
Filename:	s:\projekte\117_erm\ser\ddcspc\ddc100ae.doc/WMO	
Synopsis:	This document provides the specification of the Data Distribution Centre Sub-System	
Author(s):	....	W. Mörgeli
	....	J. Trnka
Approval:	....	Francisco Gonzalez
Distribution:	ERM	

Revision History

Version, Date	*Change Description*
Draft Version 1.0B, 17 Sep 98	USP stuff added
Draft Version 1.0A, 05 Oct 98	New Document

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to provide the description of the Data Distribution Centre (DDC) sub-system specification.

Data Distribution Centres can be considered as a kind of front-end system which have been introduced to limit the number of connected participants per system to a reasonable amount. The number of installed DDC's depends on the number of connected participants respectively their users (PAPI and traders). DDC's have been introduced to meet the non-functional requirements for ERM.

From the logical view of the business design the DDC's are not covered. The goal of this document is to extend the business related design to cover the technical and the business requirements on the DDC's.

Before you start reading this document you should be familiar with the specification of ERM architecture document I-ERM-SAD.

1.2 Scope

The scope of this document covers the Data Distribution Centre sub-system.

1.3 Definitions and Abbreviations

API	Application Programming Interface
EBS	Elektronische Börsen Schweiz
ERM	Electronic Repo Market
ES	Exchange System
FS	Functional Specification
IOI	Indication Of Interest
MAC	Message Authentication Code
POA	Participant's Own Application
RTR	Reliable Transaction Router
SECOM	SEGA-Communication
SMF	Swiss Market Feed
TS	Trading System
TCP/IP	Transmission Control Protocol/Internet Protocol
NTB	Network Transaction Bus

1.4 References

reference & document title	**applicable version and reference**
🗎1 ERM Software Architecture	I-DEV-SAD-100D/D
🗎2 ERM Systemkonzept	I-DEV-ERM-100D/D
🗎3 Functional Requirements for ERM	I-ERM-FRQ-100A/E

reference & document title	applicable version and reference
🗎4 Java Feasibility Study	I-ERM-JFS-100
🗎5 EBS Project High Level Design (ES)	I-ESD-HLD-103/E
🗎6 Gateway Interface User Guide	I-ESD-GUT-106/E

1.5 Outstanding Issues

Currently none.

2. Overview

This chapter summarises the overall architecture of the ERM system. The detailed description can be found in 🖹7.

The major components of the ERM system consist of the following components:

- Participant Application API (PAPI)
- The Participant Own Application (POA)
- Trader Graphical User Interface (GUI)
- Internet Connection Interface (USP Servers)
- Independent Data Distribution & Recovery Centres (DDC’s)
- Host System (Business Centre)

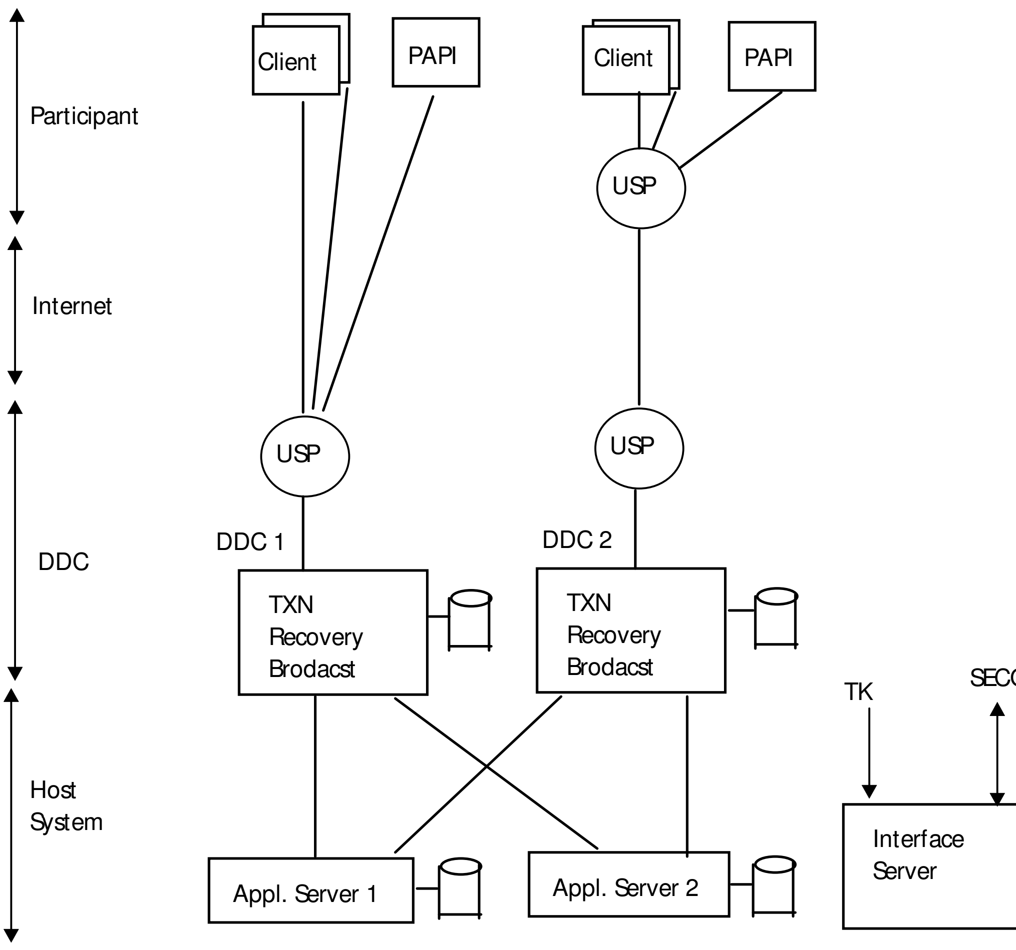
The Graphical User Interface (GUI) is a JAVA based application presenting the business information to the traders and provides the required forms, for example, to enter order requests. The GUI application is in the position to connect via Internet to the Universal Server Control Process (USP).

The PAPI enables the members to perform computer based trading by sending order requests and receiving ES feed information (trades, orders...)

The Universal Server Control Process (USP) enable the GUI applications to connect to the transactional and the recovery processes residing on the Data Distribution Centres. On the other hand the USP provides support to broadcast messages from the DDC’s to the connected clients.

The Data Distribution Centres maintain replicated information from the host databases and provide services for transactional messages, message recovery and broadcasting of market data.

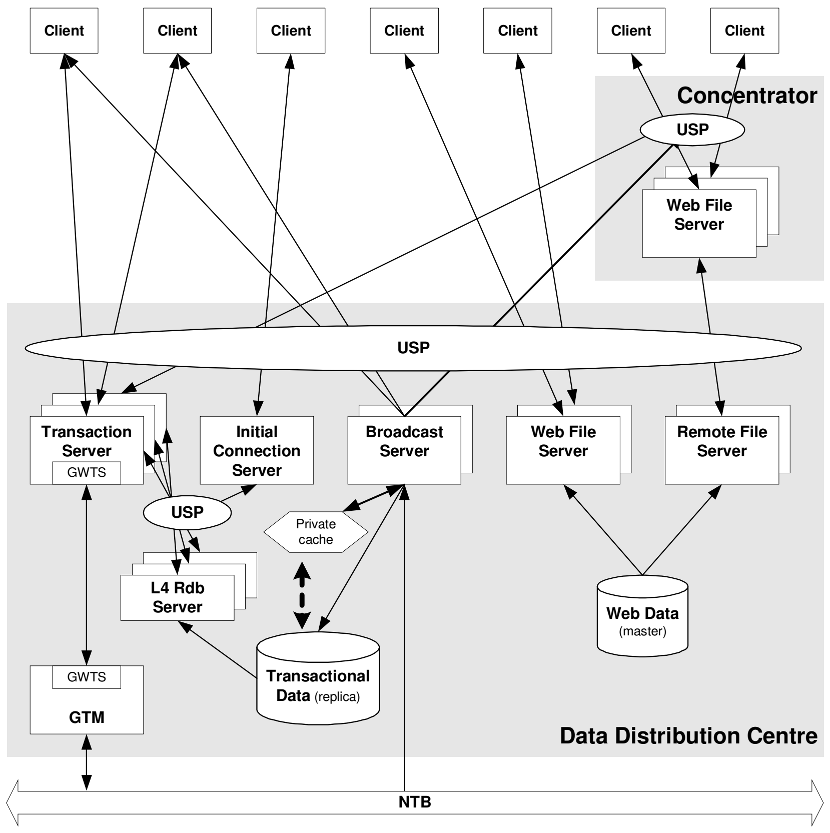
The business applications reside on the host system, i.e. the order matchers, which maintain the master copies of the business databases. In addition, the host system houses the Common Business Control (CBC) database which contains the static configuration data and the information that is common to all markets.


Participant
Internet
DDC
Host System
Client
PAPI
Client
PAPI
USP
USP
USP
DDC 1
DDC 2
TXN Recovery Brodacst
TXN Recovery Brodacst
TK
SECO
Interface Server
Appl. Server 1
Appl. Server 2

3. Architecture

3.1 Overview

The following schema shows the overall architecture of the DDC.



Before a session connection is established, the Initial connection server redirects the client to the appropriate access point. Access Point can be a Data Distribution Centre or a concentrator.

Client access to static (HTML or Java) or dynamic (generated HTML) Web data (non-transactional) is provided by local or remote Web File Server.

The clients are permanently connected to a instance of a transaction server and receive messages from one or multiple broadcast servers.

Transactional data is stored in a local replica of the database. Read only requests are covered directly by the transactions server in co-operation with the LEVEL4 Rdb server. Request for data modification are passed to the Transaction Dispatcher Process GTM and further via NTB to the business server.

The broadcast server maintains a private data cache to supply information to the clients as fast as possible.

Multiple Data Distribution Centres coexist in one cluster or as separate nodes and bear the load of up to 200 clients each. The performance of the system increases with the number of DDC's and with number of concentrators.

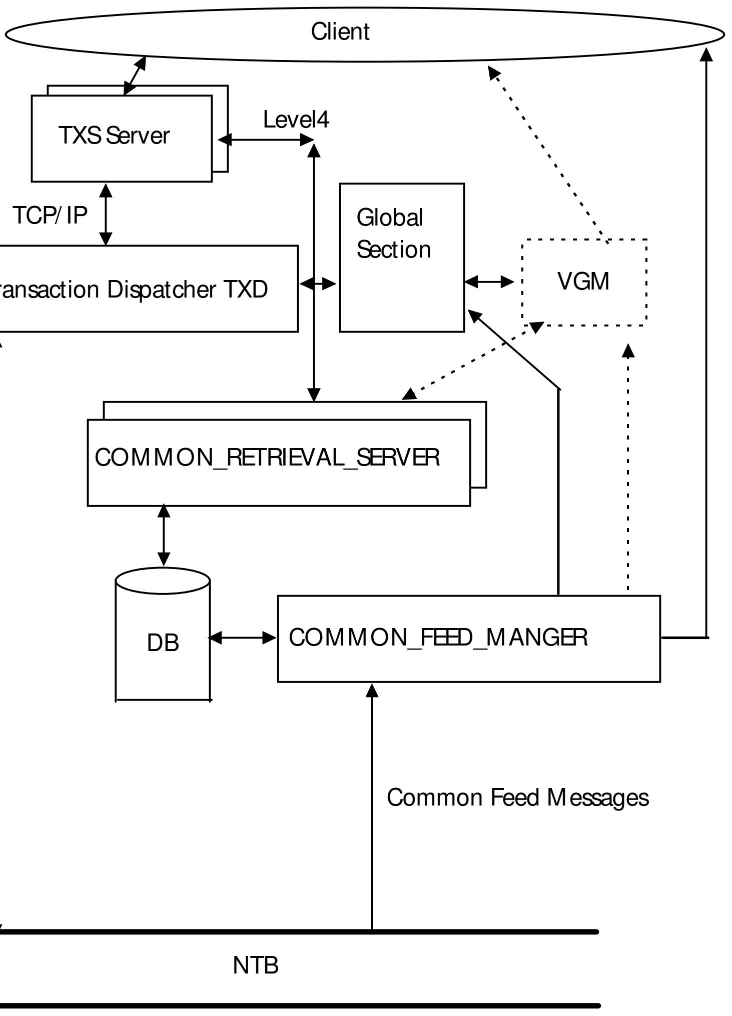
The primary purpose of the concentrator is to provide local Web services for static data and to distribute (fan-out) broadcast messages to connected clients. Transactional services are pipelined between the USPs.

3.2 DDC Process Configuration

The process configuration on the DDC is split into two area's. The first area covers the configuration to process the data stored in the common area (CBC database) and the second area focus on the market specific process layout.

3.2.1 Common Process Configuration

This section provides an overview of the process configuration for the DDC common



The Transaction & Context Server (TXS) process is created by USP for every connected client application which means that there can be approx. up to 300 instances per DDC.

The Transaction Dispatcher (TXD) is instantiated as a single process per DDC and serves all markets with transactional messages by referencing the NTB transport layer.

The COMMON_RETRIEVAL_SERVER serves request from the GUI's or the VGM for data such as:

- trader profile information
- static instrument information
- change fixes

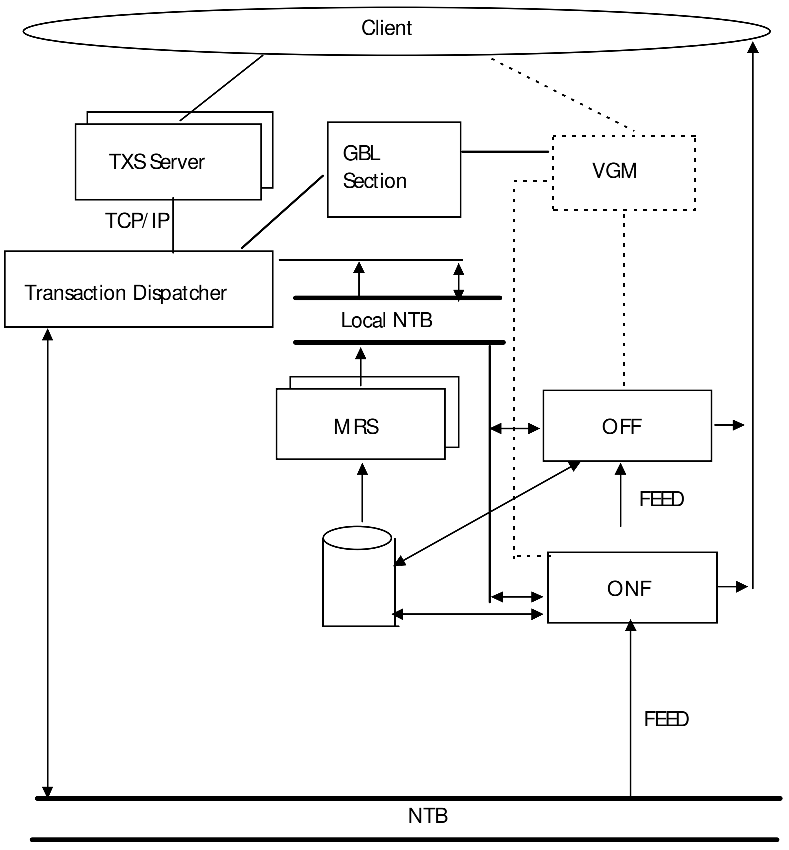
The COMMON_FEED_MANAGER processes the feed from the operations applied to the Common Business Control (CBC) database. A sub-set of the messages are disseminated to the GUI's via USP. VGM needs to be notified about any limit change applied to the CBC database.

The initial limit data is requested by the VGM during start-up from the COMMON_FEED_MANAGER process for synchronisation reasons.

The global section will contain the list of connected participants together with the market specific USP broadcast mask and a reference counter with the number of users connected from a particular participant. There might be a need to also store the global limit values – to be decided.

3.2.2 Per Market Process Configuration

This section provides an overview of the per-market process configuration.



OFF_OFF_MARKET_FEED_MNGR

This process reads the off-market feed and updates the local database. The connected clients are notified by a broadcast message via USP. The open IOI's and ADO's are kept in an in-memory database for fast retrievals for the client applications requesting this information on the DDC local transport facility.

ONF_ON_MARKET_FEED_MNGR

This process reads the on-market feed and updates the local database. This includes also the processing of the market trades. The clients and the VGM are informed about any new or modified order. The orders are kept in an in-memory database for fast retrievals for the client applications and the VGM.

MRS_MARKET_RETRIEVAL_SERVER

This process operates as a retrieval server for client requests to get market specific information such as opened/closed trades (public or own) according to a specified time window.

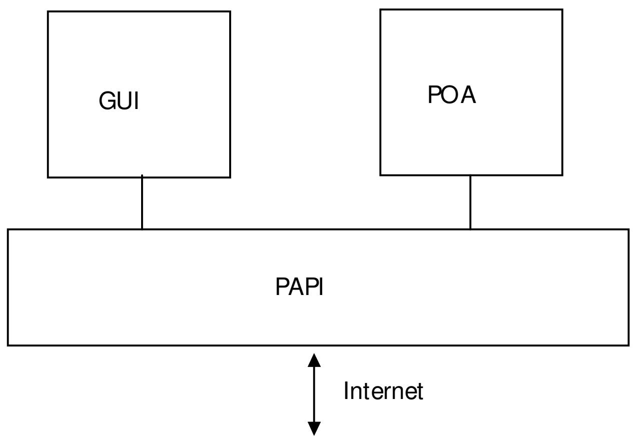
Issue:

Sine there are market specific EUI applications there might be a need to install a Market Business Control process to receive the messages in order to synchronise the Common Control Database.

3.3 Participant Application Programming Interface (PAPI)

The Participants Own Applications POA’s using the Participant API (PAPI) will have the services available which are provided for a standard GUI application. That is, POA and GUI applications are based on the same software layer and thus can share the same functionality.

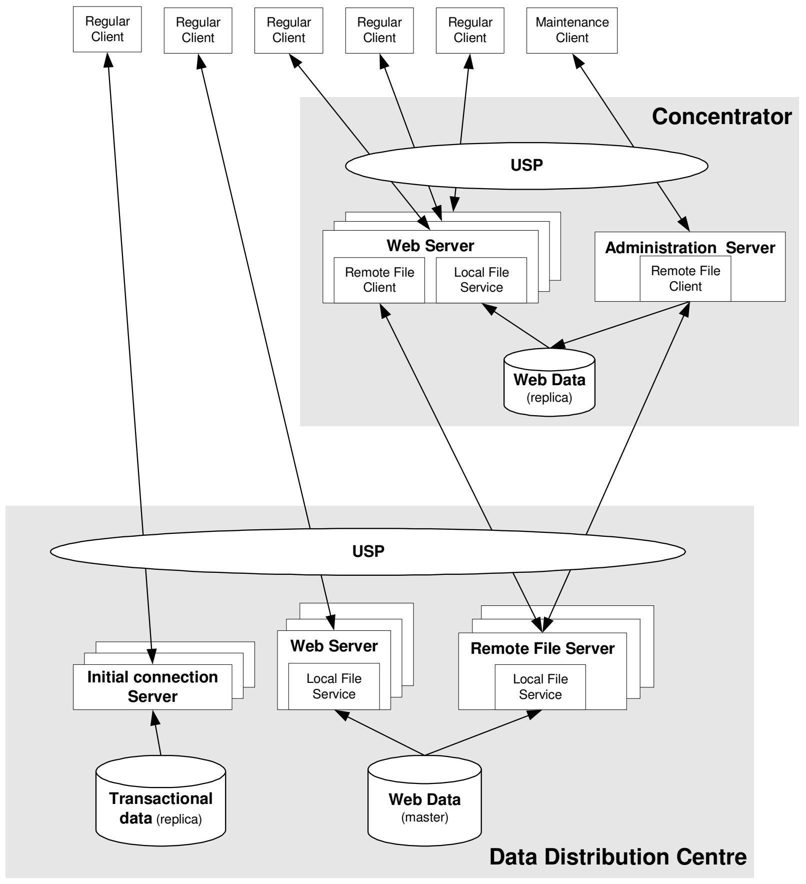
No extra functionality will be available for the POA !



4. Transport Protocols

4.1 Web Connection

The connect the client to the DDC initial web connection is required re-direct the client to the nearest Access Point, to load static HTML pages and Java classes. The following schema shows these components.



4.2 Initial Connection Server

Before a client starts working a web connection must be established to load required Java classes into the browser. For security reasons Java allows to access only nodes from which the classes has been loaded. For the ERM system the DDC or the Concentrator are the access points from which theses components can be loaded. The purpose of the initial connection server is to fine the best suitable access point for this client.

Based on the client code (e.g. UBS_HARTMANN) passed in the URL request, the server retrieves the configuration DB and finds the nearest available access point for this client. It generates a HTML with a URL re-direction and sends this page back. The client browser or the Java application interprets this information and sends a URL request for the HTML start page to the access point. The re-direction is done with the standard HTML tag:

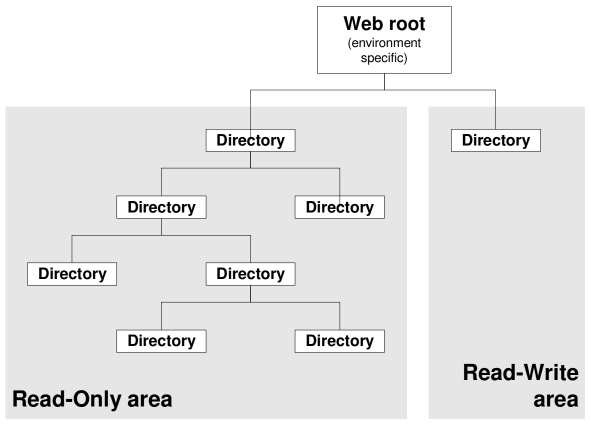
```
<META HTTP-EQUIV="REFRESH" CONTENT="0;
URL=http://www.accesspoint.ch/T45/start_page.html">
```

Additionally the initial connection server verifies the IP address of the client against the database.

The server is running within the USP and OMC control as a stateless server. Multiple instances are pre-started to support concurrent client connections.

4.3 Local Web File Server

The local web file server provides extended file services to the client. It delivers static and dynamically generated HTML pages, images, and Java classes to the client. The files are stored in a directory tree under a common root. One part of the tree holds all static data, the other one (flat directory) holds temporary generated data.



The server deletes files with a specific file name pattern (e.g. all files matching the pattern %%%_TEMP_*.HTML) after successful send.

The server is running on any DDC and on any concentrator. It serves all environments on the particular node. The environment switch is driven by the first parameter after node in the URL.

```
http://www.accesspoint.ch/T45/struc/test/mypage.html
```

In the example above the ULR maps to the file:

```
VSI_ENV_ROOT:[T45.WWW_ROOT.struc.test]mypage.html
```

On the concentrator the local web file server has no access to Dynamically generated HTML pages which are stored on DDC. For this reasons it can connect to a remote web file server on the DCC and get all necessary data from there.

The server is running within the USP control as a stateless server. Multiple instances are pre-started to support concurrent client connections.

4.4 Remote Web File Server

The remote web file server extended the functionality of the local web file server. On DDC this server provides file services to local web file servers running on Concentrators for dynamically generated HTML pages stored on DDC.

On concentrators and DDC it additionally provides file services for administration server to maintain the replicas of static web data.

The file storage on the local and remote node must be the same.

The server is running on any DDC and on any concentrator. It serves all environments on the particular node. It is running within the USP control as a statefull server. Multiple instances are pre-started to support concurrent requests.

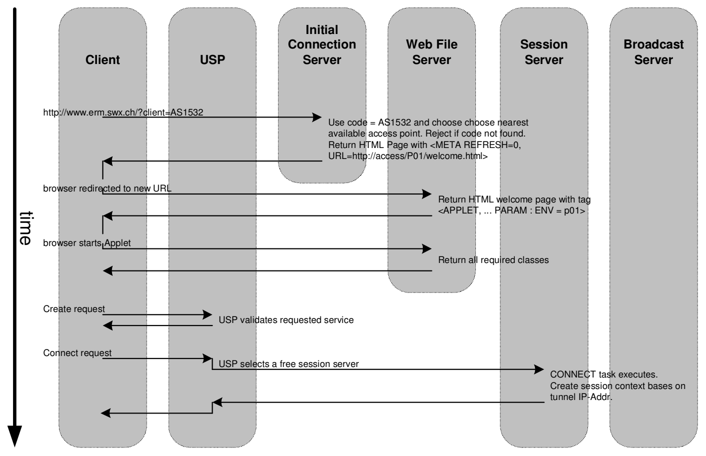
4.5 Web Administration Server

The purpose of the administration is to simplify the maintenance of the web data. It helps the administrator to distribute replicas of static web data to the concentrators and to DDCs. It provides an overview about the stored data and their structure and allows a simple file manipulation.

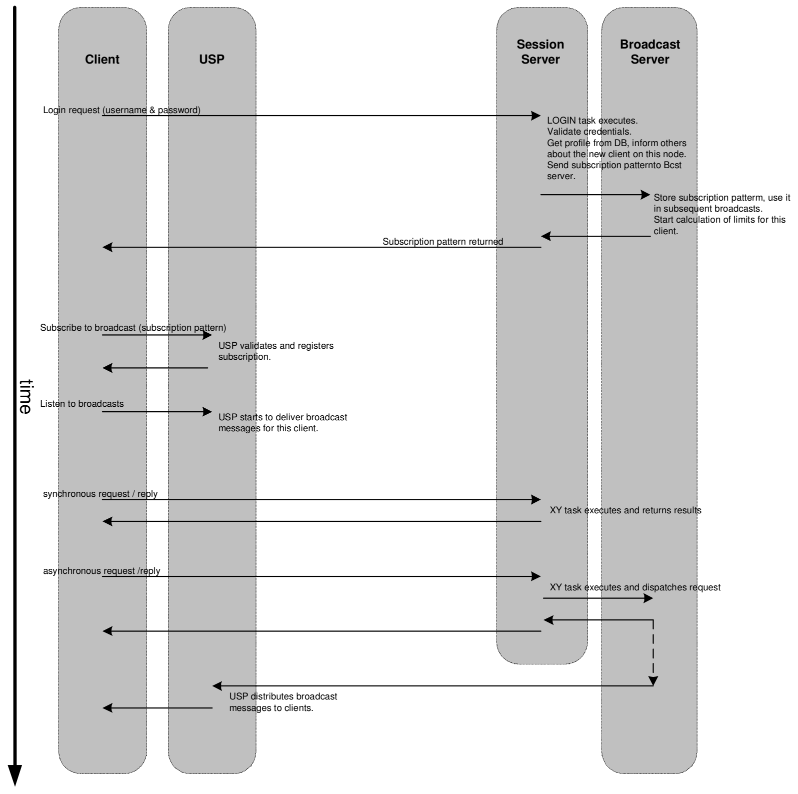
The server is running on any DDC and on any concentrator. It serves all environments on the particular node. It is running within the USP control. Single instance are pre-started to avoid potential conflicts between concurrent administrators.

4.6 Initial Bootstrap Sequence

The client must undergo several states and execute multiple steps before it can successfully connect to the DDC. The following chart shows these sequence.


time
Client
USP
Initial Connection Server
Web File Server
Session Server
Broadcast Server
http://www.erm.swx.ch/?client=AS1532
Use code = AS1532 and choose choose nearest available access point. Reject if code not found. Return HTML Page with <META REFRESH=0, URL=http://access/P01/welcome.html>
browser redirected to new URL
Return HTML welcome page with tag <APPLET, ... PARAM : ENV = p01>
browser starts Applet
Return all required classes
Create request
USP validates requested service
Connect request
USP selects a free session server
CONNECT task executes. Create session context bases on tunnel IP-Addr.

Once the Client is connected Session Start-up sequence is executed to authorise the client, load static data and establish initial business conditions. The following schema shows this sequence.

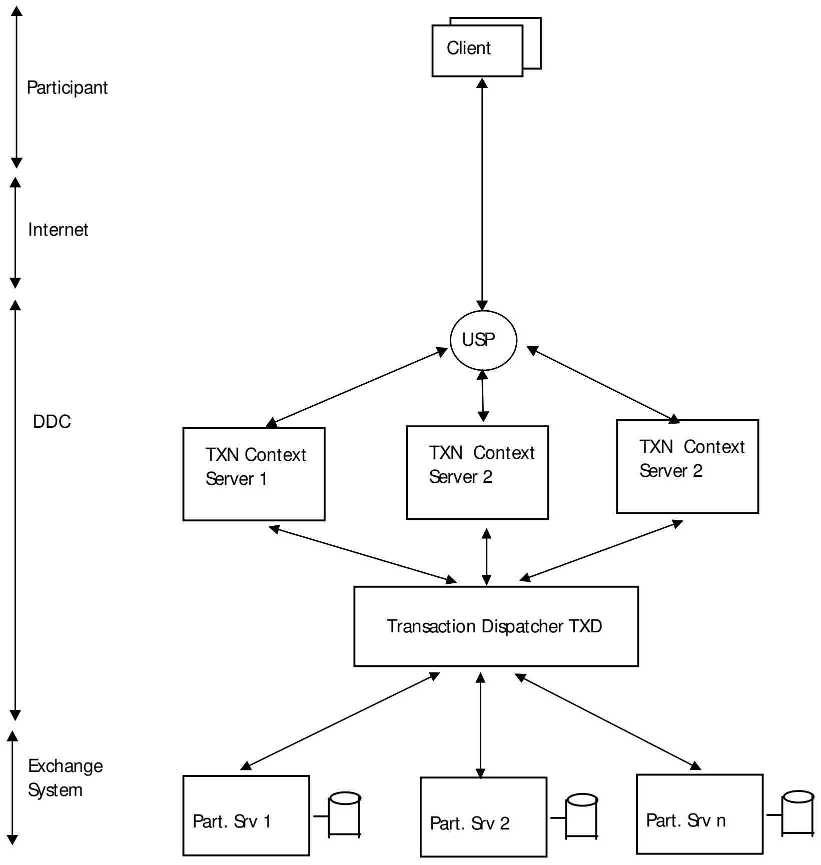

time
Client
USP
Session Server
Broadcast Server
Login request (username & password)
LOGIN task executes. Validate credentials. Get profile from DB, inform others about the new client on this node. Send subscription patternto Bcst server.
Store subscription patterm, use it in subsequent broadcasts. Start calculation of limits for this client.
Subscription pattern returned
Subscribe to broadcast (subscription pattern)
USP validates and registers subscription.
Listen to broadcasts
USP starts to deliver broadcast messages for this client.
synchronous request / reply
XY task executes and returns results
asynchronous request /reply
XY task executes and dispatches request
USP distributes broadcast messages to clients.

From now the client is able to perform business functions.

4.7 Transactional Messages

4.7.1 Host Application Messages

The data flow of the client initiated transactional business messages is shown in this diagram.



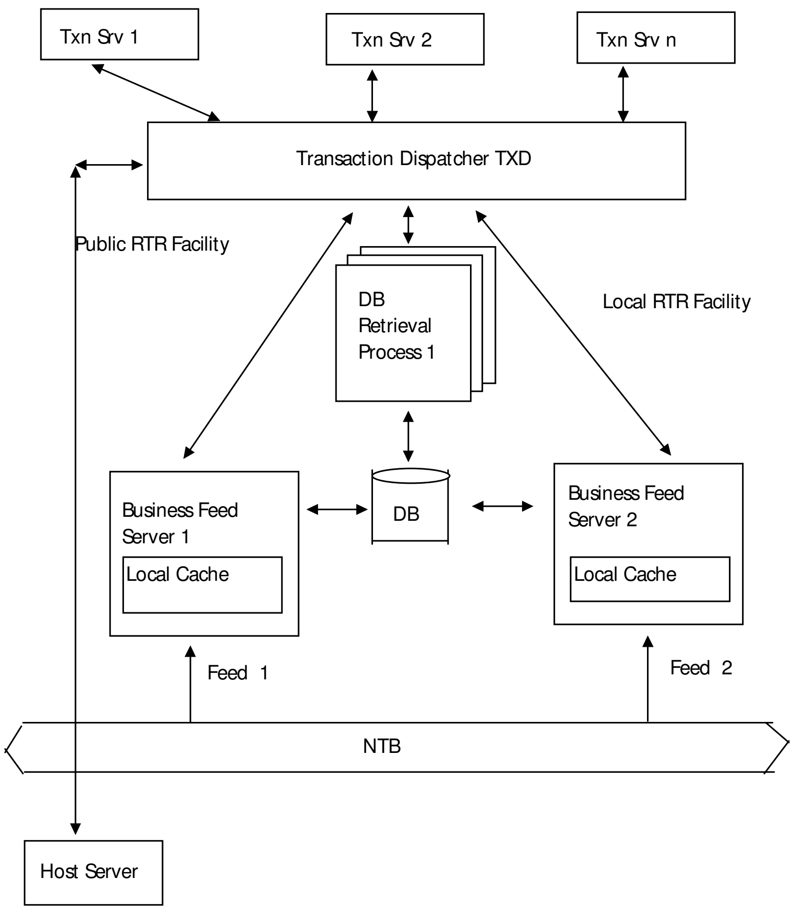
A transactional client request/reply message is processed the following way:

1. An initial request message is sent by the client application via USP to the related context server TXS.
2. The context server TXS forwards the message via TCP/IP to the Transaction Dispatcher TXD.
3. According to the market code the message is sent via NTB by the Transaction Dispatcher TXD process to the related business server process by using a market specific NTB client channel.
4. The business process may respond with a single or with multiple reply messages which are fed back to the client over the established connections.
5. Optionally the client might send several request messages in sequence depending on the business case. The message type indicator in the header defines the protocol type.

4.7.2 Data Retrieval Requests

All data retrieval requests need to be satisfied in ERM by the DDC's in order to off-load the host system. To benefit from the built-in NTB routing mechanism these requests are processed by a DDC specific RTR facility which has the front-end, router and backend rules defined on the local system only. Any RTR configuration spawning backend servers on multiple DDC's could result in significant system instabilities according to the experiences made in the EBS project.

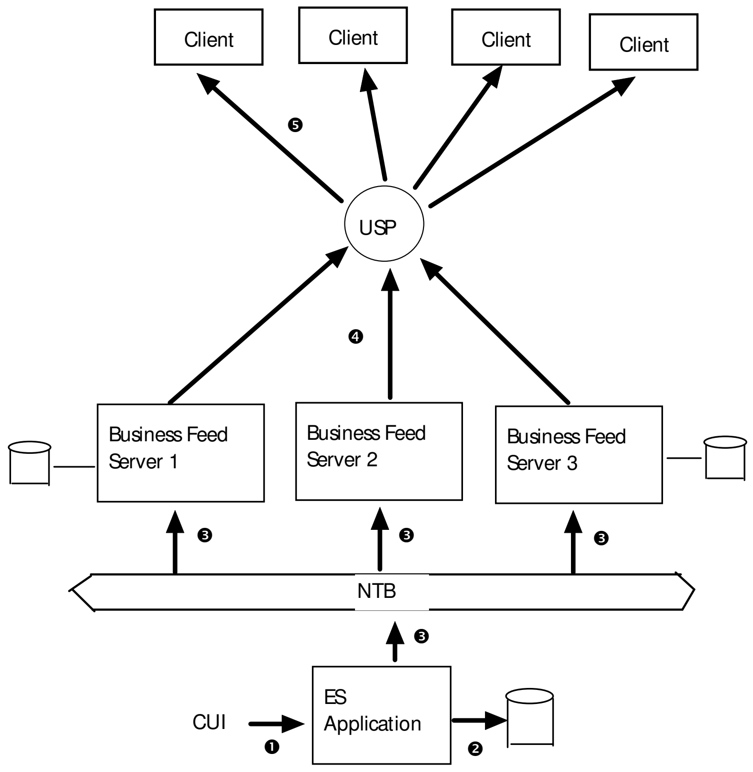
Data retrieval requests can be routed to the implemented feed server processes or to dedicated data retrieval processes for those cases where a large amount (i.e. trades) need to be retrieved. The retrieval processes may be installed as concurrent server processes to satisfy simultaneously client requests.



4.7.3 Broadcast Messages

Broadcast messages are released by the business applications running on the hosts when there is a business need to notify the connected clients in a market or to safely disseminated messages within the host systems (i.e. auditing).

The DDC applications are not sending broadcast messages.



In the above diagram the following scenario could be applied:

1. The CUI application sends a message to a business application. I.e. a trader suspension message.
2. The business application updates the database
3. A broadcast message is disseminated via NTB to the Business Feed Server processes on the DDC.
4. The message is formatted into a client compliant format and forwarded via TCP/IP to the USP broadcast service.
5. The USP server broadcasts the message to all connected client applications which previously had subscribed to the related broadcast service.

5. Concept of Virtual Displays

To minimise the recalculation of the information displayed on the GUI's virtual displays need to be maintained and distributed by the DDC's. Since there is no means to safe-store data on the client side every request to get the information to be displayed needs to be retrieved from the DDC.

As an example, the inside market could be calculated by the on- and off-market business feed server applications and data updates simply could be disseminated to the clients via the USP broadcasting mechanism.

To avoid synchronisation problems between a snapshot request and a broadcast message the DDC's must NOT send any delta values, i.e. subtract n from a given volume. ALL values distributed are real values as they need to be displayed on the client side. As a consequence a concept needs to be worked out to identify each displayable entity, i.e. contract type or Buy/Sell and Price for the detailed order book.

An initial request to get the complete information to be displayed would be implemented as follows:

- The client sends an initial request containing the object identification including the market id.
- The related business server process retrieves the information from the cache (or potentially from the disk) and broadcasts the displayable entities on the participant private data stream.

Consolidated information update messages delivered to the client could be sent periodically by the DDC's, i.e. every n seconds, to efficiently use the reduced Internet network bandwidth. Fairness is achieved by distributing the messages on an absolute time basis assuming that the system times on the DDC's can accurately be synchronised by DTS (deviation < 0.5 second). In case of a saturation of the network bandwidth the send interval time could be adjusted accordingly.

6. Support for Multiple Markets

There is a functional requirement to support multiple (independent) markets within the Repo project. Because of:

- scalability
- different trading hours in each of the markets
- operational aspects (backups)

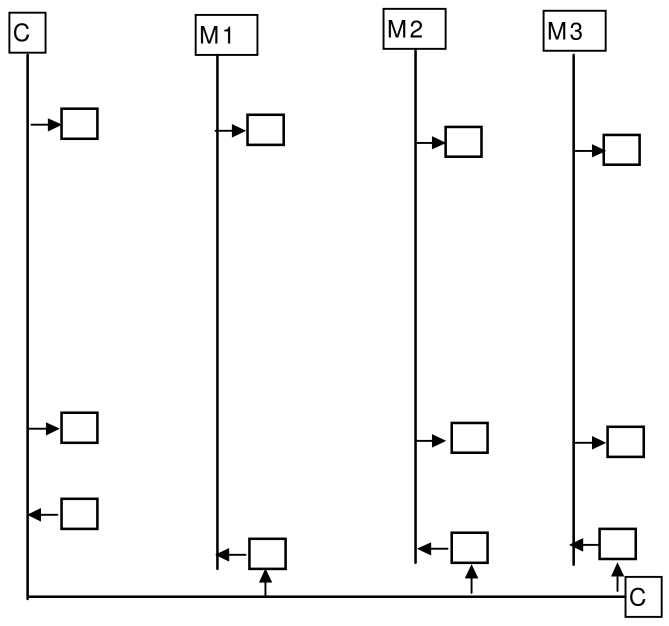
To be in the position to address all markets from a single application instance, i.e. the Transaction Dispatcher (TXD), or to exchange messages between the different markets shared RTR facilities will be implemented for all markets.

This means that the NTB routing key information needs to be unique for all markets. This is the case in EBS where each of the tradable entities is identified by a unique security id. In the Repo project it has been confirmed by GBM that the contract identifiers are unique for all the different markets. This concept will further support to partition business applications within a single market whereby each of the partition processes a sub-set of the market assigned tradable entities.

In order to avoid that broadcast messages released by business applications in one market are not distributed to receiver applications in other markets a market specific RTR filter mask will be implemented.

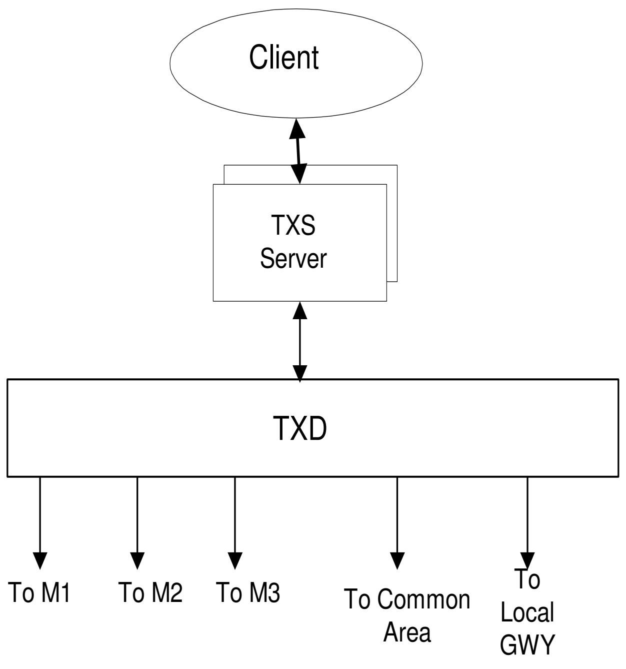
6.1 Market Relay Servers

To exchange Store/Forward messages between different markets the concept of so called Market Relay Servers (MRS) has been introduced. These processes listen on the Store/Forward channel in one market and send the received messages on the Store/Forward channel of a different market.



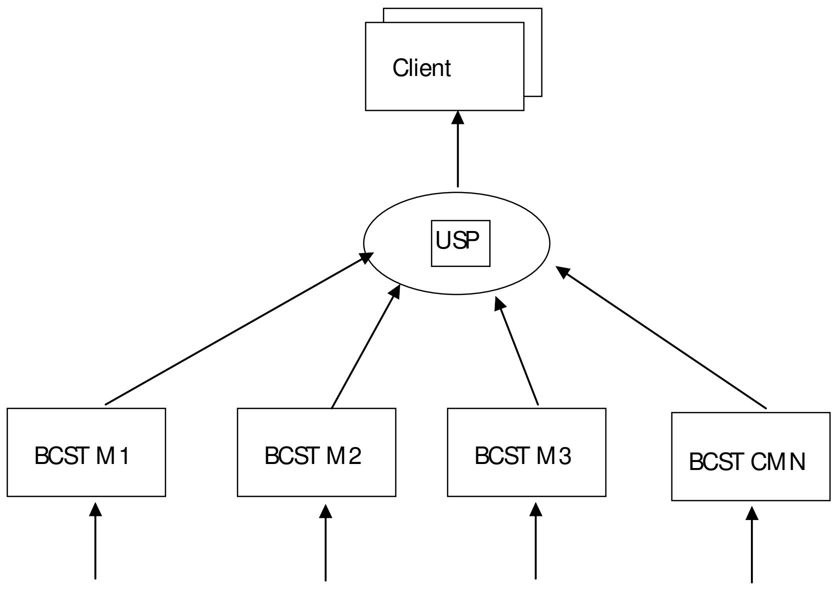
6.2 Transaction Dispatcher (TXD)

In a multi-market environment the Transaction Dispatcher (TXD) responsible for routing transactional messages is an exception in the sense that there is a single instance per system only which serves all markets including the local DDC.


Client
TXS
Server
TXD
To M1
To M2
To M3
To Common
Area
To
Local
GWY

6.3 Broadcasting via USP

The Universal Server Process (USP) can be considered as a layered product. Broadcast message from all markets are distributed via the USP facility.



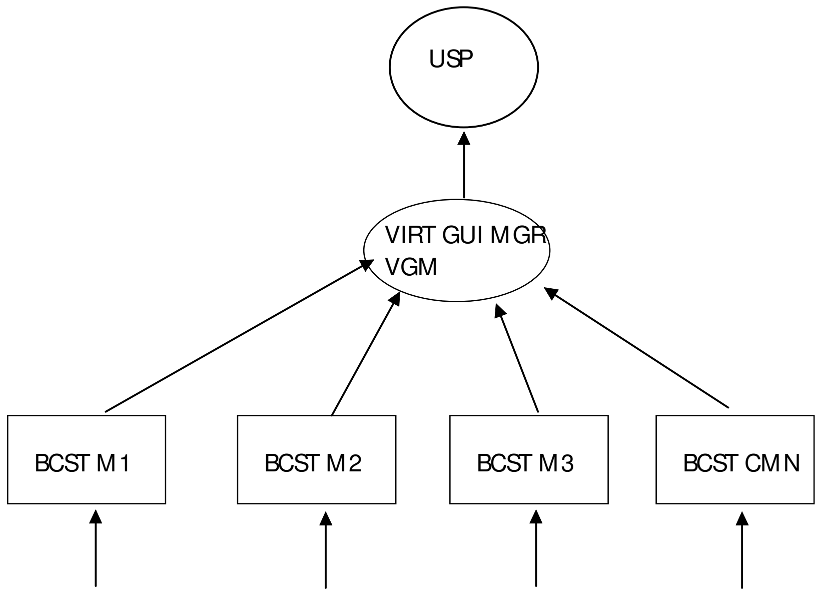
6.4 Distributing Market Limit Data

The limit information is common across all markets**.** To meet this requirement it is proposed to implement a common Virtual GUI Manager (VGM) on the DDC's where potential synchronisation problems on the data feeds can be sorted out. The current available limit value is calculated based on the initial limit and the open trades per member. For performance reasons this process maintains the virtual GUI's only for the connected members. To get a consistent view when a member connects during operation and the start-up initialisation could be sorted out as follows:

- At start-up the VGM gets a snapshot from the limit database GLM including the current sequence number defined per member pair mx – my
- Each update in the GLM database caused by a trade forces a sequence number to be increased which is allocated per limit value
- The VGM recalculates the available limit after receiving a trade. To avoid synchronisation problems between the different markets it is crucial to compute the values based on a delta (difference) and not on an absolute value. The limit sequence number shipped with each trade is compared with the update number received in the snapshot. If it is higher we are fine. If it is lower the local limit value is adjusted according to the delta value received – plus when the delta value will be subtracted, minus when added later again.
- The same logic applied when a limit is adjusted by a participant during trading. The new limit value needs to be accompanied by the update sequence number as well and it is important that also the difference of the limit before and after the update is known enabling the VGM to recalculate the correct value.

Notice:

Since the limit values as well as the traded amount of money can be in any currency a change-fix rate needs to be referenced and a recalculation needs to be performed on demand.

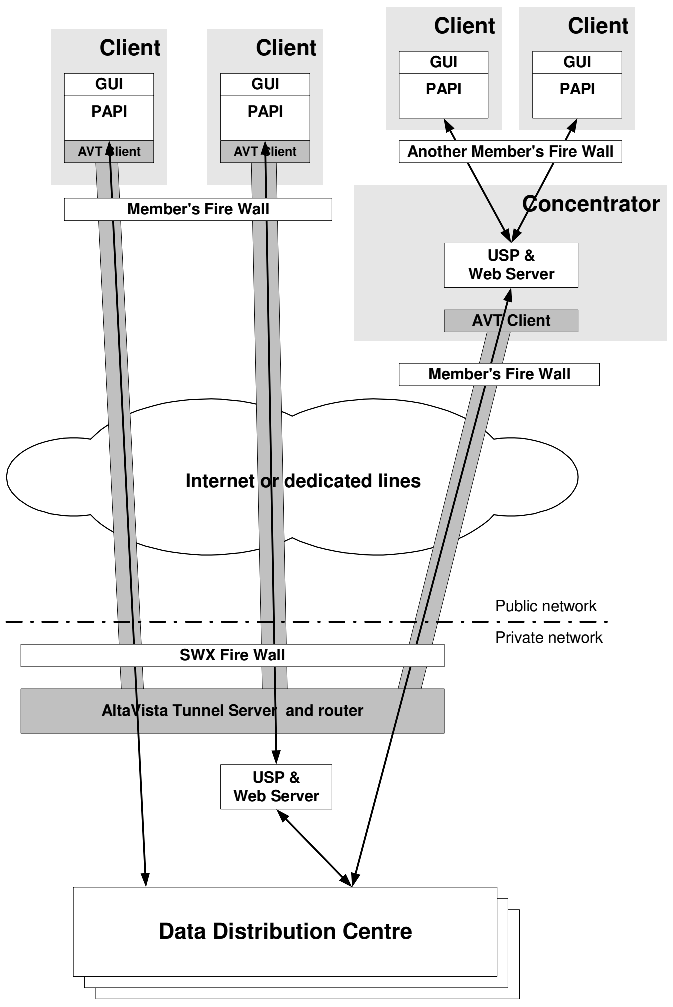


7. Security

The security concept is based on several components:

- Authentication of the communication partners
- Encryption of the transferred data
- Encapsulation of the network traffic
- Authorisation of the client
- Server based session context
- Client Privileges

The next chapters explain each of these components if detail. The schema below shows how clients are connected to the system.



The client or the concentrator are connected to the system through AltaVista tunnel (AVT). This product provides encryption, encapsulation and authentication on top of the network protocol. This product type is certified by NSCA and ITSEC

7.1 Authentication

Authentication ensures that parties are sure to communicate with the right partner. The AltaVista Tunnel provides authentication function. Each AVT-Client uses its own unique pseudo IP-address assigned by the AVT-Server. This address is verified by the initial connection server in the database on the DDC and connections with invalid IP address are rejected.

7.2 Encryption

Encryption ensures privacy of data transferred between the parties. The AltaVista Tunnel provides encryption of all data. It is based on the known principles of RSA technology. The key length is 128 bit and changed periodically (default interval is 30 minutes). Several encryption algorithms can be used.

7.3 Encapsulation

Encapsulation ensures the application network protocol is not as such recognisable from the outside world. E.g. the protocols like HTTP, FTP, TELNET or even private protocols are hidden and can not be subject of an attack from parties not participating in a tunnel connection. AltaVista Tunnel provides encapsulation of any TCP/IP based network protocol.

7.4 Client Authorisation

The AVT Client ensures that only authenticated clients can connect to the DDC. The client authorisation is done on the application level. The client passes client short name and password in log-in request which must be executed as the very first action after the client is connected. They are both verified in the database. Any further RPC requests without an initial successful authorisation are rejected. This mechanism prevents the system from being intruded

7.5 Server based session context

Since the connection between the Client and the DDC is established, the transaction server maintains the context of the client. The context contains information about the client authorisation, about the participant permissions to execute functions and about the state of the connection. Any further operation is executed in this context.

7.5.1 Client ID

The client ID is generated by the ERM system when new client is created and stored in a central database. It is unique over the entire system and is a part of the session context hold by the transaction server. For security reasons the client ID is not disclosed to the client itself. For ongoing identification the client has a short name and full name both unique within the participant.

7.5.2 Participant ID

The participant ID is generated by the ERM system when new participant is created and stored in a central database. It is unique over the entire system and is a part of the session context hold bay the transaction server.

7.5.3 Participant broadcast mask

USP provides the ability to broadcast messages from the DDC to the connected clients. Via a so called broadcast mask the range of clients to be addressed by a single send operation can be limited.

The requirements are:

- to send broadcast messages to all participants in all markets
- to send broadcast messages to all participants in a dedicated market
- to send broadcast messages to all clients of a participant
- to send broadcast messages to a specific client of a participant (this is not a business requirement, but a technical solution to optimise start-up processing)

Broadcast mask layout

Attribute	Format	Comment
Market_Mask	X(1)	Single character market specific mask. Example: R for Repo
Participant_Mask	x(6)	Participant specific
Client_Mask	x(4)	Client specific

Examples:

'*	' addresses all clients in all markets
'R*	' addresses all clients of all participants in the Repo market
'R_12*	' addresses all clients in the Repo market of participant identified by 12
'R_12_1234	' addresses client 1234 of participant 12 in the Repo market.

7.6 Participant Privileges

The maintenance of participants is under the control of the Swiss Exchange. When participants are created, number of authorities must give the approval. Participants are authorised to take part in a market and to play a role within them.

7.7 Client Privileges

When new participant is added at least one client must become administrator rights. This authorise the client to create, modify and delete additional clients of this participant.

8. Diagnostic Messages and Codes

This section describes the rules about diagnostic and validation codes delivered by the DDC to the client. Diagnostic codes reflect the technical state of a transaction and the validation codes are business related.

8.1 Infrastructure Related Conditions

The technical diagnostic code indicates the outcome of the transaction whether the operation physically could be completed or not. There are three possible codes:

- success meaning that the transaction technically completed successfully
- failure meaning that the transaction definitely failed
- unsafe meaning that the DDC does not know the exact outcome of the transaction

In the case of an unsafe condition the business transaction might have failed or succeeded. When the client receives the related broadcast message the transaction definitely has completed successfully. If not, the client needs to query the final outcome of the transaction via a DDC data retrieval process.

When the transaction server dies unexpectedly then the client’s RPC will terminate with an error code (TCP/IP read error) specific to the client platform.

The final values assigned to the different diagnostic codes need to be negotiated between USP and TXS.

8.2 Business Related Condition

The business related validation code indicates the outcome of the validation done by the business application. It is usually contained in the transactional reply and in the related broadcast message.

In the case of a successful business validation code in the transactional reply message the related broadcast message will be disseminated to all DDC’s and the client will also be notified by an ‘object’ broadcast message.

In the case of a validation failure returned in the transactional reply message a related broadcast message will be sent to all DDC’s and stored there for later client query requests. NO ‘object’ broadcast message will be released to the clients in this case.

The rules for the validation codes are:

- 0 means success
- not 0 means that a business validation check has failed

It is important to note that the technical diagnostic code described in the previous section is set to success independent of the validation code defined by the business application (still assuming that no technical problem occurred).

9. System and Process Configuration

9.1 General Resource Names

To cater for the requirement to have multiple environments, each running multiple products with multiple markets the proposed standard for resource names is:

<env>_[<product_id>]_[market_id]_<identifier>

A product_id would for example refer to R for Repo and E for EBS. Market_id's could be CH (Swiss) or GL (Gilt).

The product_id and market_id will be defined by VSI or defined by OMC at startup of an application in form of logical names in the process logical name table or retrievable by an API call.

Proposed names:

VSI_ENV_PRODUCT_ID	for the product_id
VSI_ENV_MARKET_ID	for the market_id

Configuration symbols or logical names required for processes running in a system should be prefixed by VSI_ENV to avoid a conflict with other VSI symbolic names.

9.2 Transaction & Session Context Server

A free instance of this process is chosen by USP each time a client connects to USP. Many instances can be pre-started to speed up connection of multiple clients at the same time. This server maintains the session context for each connected client and communicates via TCP/IP with the transaction dispatcher process.

There would not be an immediate need to register these context server processes with OMC. Registering with OMC results in the following advantages:

- OPS is in the position to monitor the instances from ESMAN
- OPS may force these processes to shutdown in emergency cases
- Statistic value can be collected via the STATS API.

The disadvantage is the increased start-up time because in the current version of the runtime environment EBS Log-in is executed.

Due to the fact that there is a significant number of these process instances running on a single DDC (~200), the application instances need to be pre-configured in the OMC database. That is, OMC needs to be informed about the maximum number of supported instances running on all DDC's (~1000). From the server point of view, this configuration information is transparent for the transaction & context server processes and no special processing is required.

9.3 Transaction Dispatcher Process

There is a single instance of this process on a DDC which has a TCP/IP link to each of the transaction & context server processes. Its main purpose is to send the messages via NTB to the appropriate business process.

A business process may either be operational :

- in the common area on the host
- in a dedicated market on the host
- in the common area on the DDC
- in a dedicated market on the DDC

To cater for this requirement there is an approach to establish a session facility per market (whereby the common area on the host is considered to be market 0) and a local facility to communicate with the DDC common and market specific retrieval processes on the DDC's. In practice all market facilities could map to the same physical RTR facility in a first phase.

We presume to get the channel descriptor information from OMC. Detailed mechanism needs to be worked out...

The maximum number of RTR client channels per process is in the current implementation limited to 255. The number of client channels per market is OMC configurable.

9.4 DDC Business Process

In EBS there has been a need to access a single database per business process. In ERM there are several processes with a need to access a 2nd (or even more) database(s) which is supported by the DBIRT facility. OMC should be extended to provide names of multiple databases without compromising the EBS approach.

According to OMC a new service will be provided with an option to get the name of a database related to a user specified identifier, i.e. BUSINESS_DB_1, BUSINESS_DB_2, CONFIG_DB_1,...

9.5 USP Services

Each runtime environment needs to have configured at least two USP services:

- one service for the transactional messages which is TXS for ERM
- one service for the dissemination of the broadcast messages which is BCST for ERM

To shield the individual runtime environments the USP service name is prefixed by the environment identification resulting in the format :

<env_id>_<product_id>_<txn_service> for transactional services (e.g. T56_R_TXS)

<env_id>_<product_id>_<bcst_service> for broadcast services (e.g. T56_R_BCST)

(((The <txn_service> and the >_<bcst_service> identifiers are project specific and need to be fetched from OMC))). This allows to run concurrently multiple products in the same environment.

Development provides DCL command procedures :

- to create transaction services
- to delete transaction services
- to start transaction services
- to stop transaction services
- to create broadcast services
- to delete broadcast services

Application programmers must call the NAM facility with the NAMe_USP_SERVICE identifier to compose the complete USP service name.

Examples:

T50_R_TXS for a session service name

T50_R_BCST for a broadcast service

It has been agreed that installing USP as a layered product is the duty of OPS and configuring the USP services is part of setting up a runtime environment (DSC).

Some useful USP commands.

Examples:

```
$ LEVEL4/MONITOR
```

```
LEV$Monitor> START SERVICE T50_R_TXS

LEV$Monitor> STOP SERVICE T50_R_TXS

LEV$Monitor> DIR/SERVICES

LEV$Monitor> HELP
```

9.6 TCP Port Numbers

In ERM there is a need for a single TCP service used by the transaction & context server to connect to the transaction dispatcher process. To allow to run concurrently multiple products in the same environment the product id is mapped into the TCP service name.

Format:

<env>_<product_id>_<identifier>

Examples:

T50_R_TXD for Repo

A problem to overcome is the current name translation for EBS which results in the format:

EBS_<env>_<identifier>

Proposed solution:

Introduce a new NAM translate function to be referenced by ERM applications only which converts TCP service names to the new standard format.

The USP uses a fixed port 9100 for communication with all clients and port 80 for Web Services.

10. Process Shared Global Section

On each DDC there is a pool of shared memory (global section) commonly referenced by all processes of all markets in a single product (i.e. Repo). This approach minimises inter-process communications and allows fast access to shared information.

The main purpose is to :

- store participant information to minimise broadcast messaging

During the participant client's authorisation phase the following information is stored in the global section:

- the market id of the connecting client
- the participant id
- the participant's client id
- the participant's client broadcast mask
- an updated counter reflecting the number of connected clients per participant

The market id together with the participant id enable the broadcasting applications to figure out whether broadcast messages need to be disseminated from the current DDC.

The broadcast mask lets an application to address a broadcast message to an individual participant.

Layout:

```
participant_id_1 .. participant_id_n
client_reference_count_1 .. client_reference_count_n
        market_id_1 .. market_id_n
        client_reference_count_1 .. client_reference_count_n
        market_broadcast_mask_1 .. market_broadcast_mask_n
                client_id_1 .. client_id_n
                client_broadcast_mask_1 .. client_broadcast_mask_n
```

n is a hard-coded value derived from the technical requirements.

11. Propagating Static Data to the DDC’s

There is information stored in the CBC (Common Business Control) which needs to be propagated to the locally maintained CBC database on the DDC’s. The EDS facility is used for this purpose and the operation is triggered on the first reception of a Start_Of_Business_Day message released by one of the running markets. (Note, that not all markets are running in the same time zone).

The business day for which the static data is fetched is stored in the database and any further Start_Of_Business_Day message received from other markets is ignored until a new business day is received which is later than the currently stored in the database.

This processing is done on the DDC by the RCFM_COMMON_FEED_MANAGER process.

Most of the queries to run on the CBC database include the BUSINESS_DAY, but there might also be queries independent of the business day.

List of tables to be downloaded and there query items can be referenced in the RCFM process specification.