SWX
SWISS EXCHANGE

Electronic Repo Market

ERM Data Distribution Center Subsystem Specification

I-ERM-DDC-100D/E, Draft Version, 29.9.1999

This document defines the decomposition of the ERM DDC Subsystem into processes and the external and internal interfaces.

Identification

Title:	ERM Data Distribution Center Subsystem Specification
Date:	Draft Version, 29.9.1999
Classification:	For internal use only
Intended Audience:	ERM Service Team
Distribution:	ERM Service Team
Keywords:	DDC, Data Distribution Center
Reference:	I-ERM-DDC-100D/E
Filename:	C:\Company_Archive\Project\SWX\REPO\I-ERM-DDC-100D.doc
Documentum:	Version: 1.1 Location: //swxdocreg/SWX WorkDB/Project/ERM/DDC/Data Distribution Center Subsystem Specification, Draft Version
Synopsis:	This document defines the decomposition of the ERM DDC Subsystem into processes and the external and internal interfaces.

Author(s):	.....	Walter Mörgeli
	.....	Jan Trnka
Reviewer:	.....	Roland Spörri
Approval:	.....	Francisco Gonzalez

Revision History:

Version, Date:	*Description:*
Draft Version 1.0A, 05 Oct 98	New Document
Draft Version 1.0B, 17 Sep 98	USP stuff added
Draft Version 1.00B, 02.11.98	Document converted to Word 97
Draft Version 1.00C, 04.12.98	Conversion and document standard errors fixed
Draft Version 1.00D, 05.07.99	Document features added in Itteration 4,5 and 6
Draft Version 1.00E, 20.05.03	New query process RTXR

Table of Contents

1. Introduction

1.1 Purpose & Scope

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.

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

1.2 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
TCP/IP	Transmission Control Protocol/Internet Protocol
NTB	Network Transaction Bus
USP	Universal Server control Process / Message Broker

1.3 References

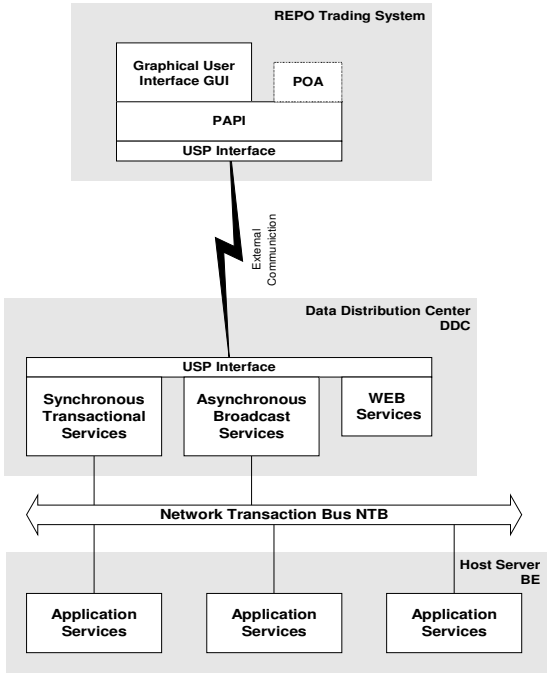
reference & document title	applicable version and reference
1 ERM Software Architecture	I-ERM-SAD-100D/D
1 ERM Systemkonzept	I-DEV-ERM-100D/D
2 Functional Requirements for ERM	I-ERM-FRQ-100A/E
3 EBS Project High Level Design (ES)	I-ESD-HLD-103/E
4 Gateway Interface User Guide	I-ESD-GUT-106/E

1.4 Outstanding Issues

None.

2. System Components

This chapter summarises the overall layout of the ERM system. The detailed description can be found in 🗎1.



The ERM system consist of the following components:

REPO Trading Client
- Trader Graphical User Interface (GUI)
- Participant Application API (PAPI)
- The Participant Own Application (POA)
- USP Interface

Independent Data Distribution & Recovery Centres (DDC's) which
- USP Interface
- Synchronous Session Services
- Aynchronous Broadcast Services
- Web Services

Host System (Business Centre)
- Application Services

The Graphical User Interface (GUI) is a JAVA based application. It presents the actual business information to the traders and allows entering all data required for trading.

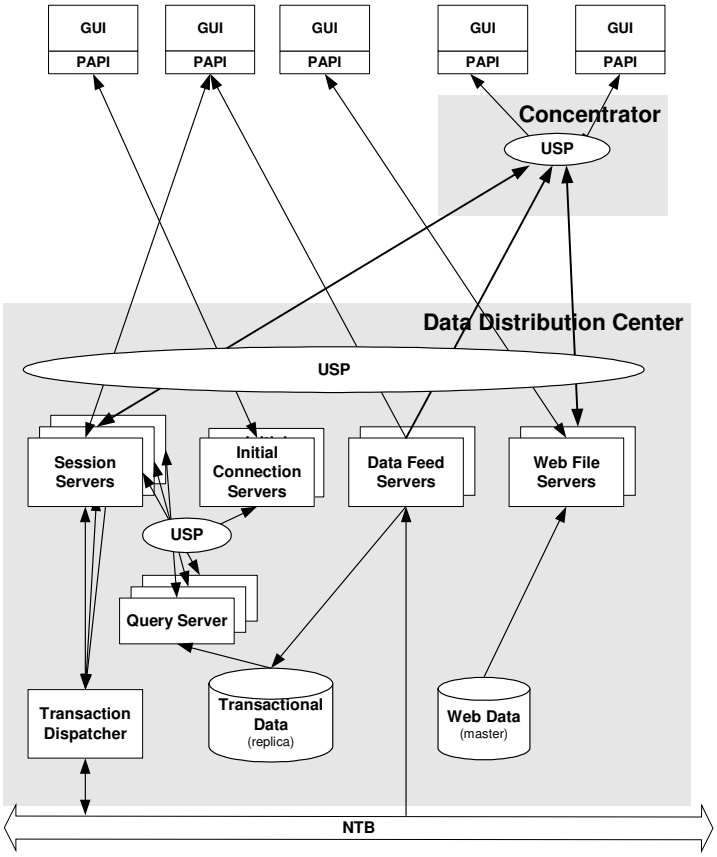
The PAPI contains parts of the business logic and a cache of business objects currently required for normal operation. It is connected to services on the DDC via an USP Interface and can on this way pass information like orders or offers for further processing. It also receives notifications, actual informations and results of the processing, which occurs on the DDC or the Host. The REPO Trading Client does not persistently store any data.

The Universal Server Control Process (USP) is a message broker with the ability to connect the trading system to the session services or Web services and also provides services for the distribution of broadcast messages to the connected clients. It consists of two subsystems RPC & HTTP for synchronous communication and BCT for asynchronous communication.

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 to this, the host system houses the Common Business Control (CBC) database, which contains the static data, and the information that is common to all markets.

The following schema shows the components of the DDC.



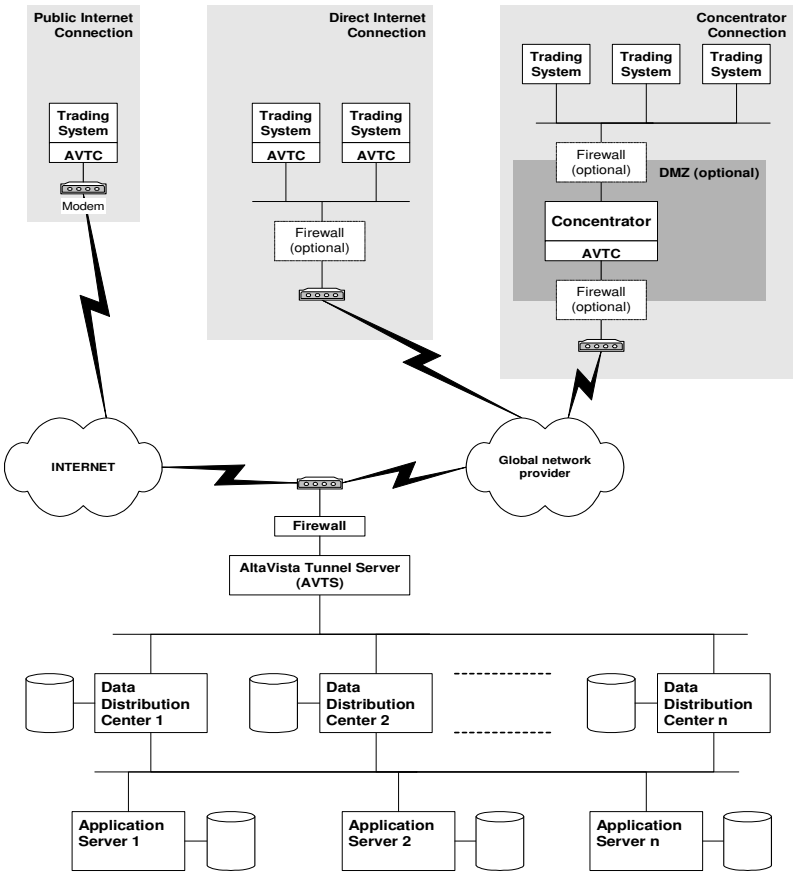
Each client is permanently connected to an instance of a Session Server and receives messages from multiple Data Feed Servers. Before a durable session connection is established, the Initial Connection Server redirects the client to the appropriate Data Distribution Centre.

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

Selected data maintained and updated by the Host System is replicated to the DDC and stored as a local transactional data in Rdb database. Client requsets for local data are covered directly by the TXS session server in co-operation with the TXR Query server. Client requests for data modification are passed to the Transaction Dispatcher Process GTM and further via NTB to the appropriate application server on the host.

The communication mode between the Client and the DDC is either synchronous or asynchronous. Both modes are described in the further chapters. 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 Nodes participating in the Cluster are called "Main Nodes".

The Client connects either directly to the DDC or to a Concentrator, which in turn is permanently connected to the DDC. The following schema shows a simplified physical connection of the components.



shown in the picture abow, there are three possible types of connection of a trading client.

1. **Public Internet connection**
   This connection type is suitable for a individual traders connected via leased line or ISDN, but through a public internet provider. The client must use WIN95/98/NT machine and the AltaVista Tunnel Client (AVTC). Depending on the location and the Internet Service Provider ISP there may be significant differences in througput and accessibility.

2. **Direct Internet connection**
   This connection type is suitable for a small number of traders of the same participant. They are connected via router (Cisco, 3COM, ZyXEL, etc.) and leased line or ISDN directly to SWX. Each client must use WIN95/98/NT machines and the AltaVista Tunnel Client (AVTC). The difference to the previous type is the privacy of the connection and the guaranteed performance of the line. The participant may protect his own LAN segments with a firewall. The maximum number of traders depends on the available bandwidth of the line. E.g. ISDN line 1x64kb = max. 3 traders.

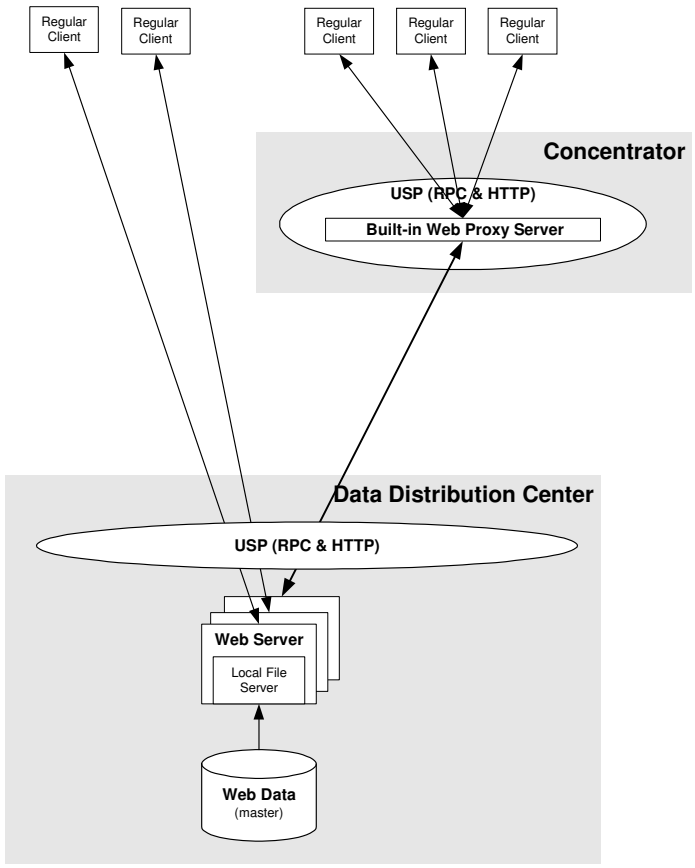
3. **Connection via concentrator**
   This connection type is suitable for participants with large number of traders (> 3). The concentrator is connected via leased line directly to SWX and provides broadcast message fan-out, traffic bundelling and web proxy services. The clients can use any types of machines (SUN, AIX, etc.), which support the Java Runtime Environment and is running the Trading Client. It does not use the AltaVista Tunnel client. This software is installed and configured on the concentrator. The difference to the previous types is the performance and scalability. The usage of a Concentrator allows establishing a DMZ (Demilitarized Zone) and so significantly improving the security. One participant may have multiple concentrators, but a concentrator is dedicated to only one participant. The security begins on the concentrator where the AltaVista Tunnel Client is installed. The participant may protect his own LAN segments with one or two firewalls. The maximum number of traders depends on the available bandwidth of the line. The usage of concentrators on ISDN lines is not recommended, because the AltaVista Tunnel Client (AVTC) needs a permanent connection with the AltaVista Tunnel Server (AVTS) on the SWX network. The usage of Concentrator is also possible for participants with public Internet connection over a high-speed leased line.

These connection types cannot be mixed at the same time. But it is possible to use a Laptop with public Internet connection during a business trip and to use the same equipment via Concentrator while working in the office later.

2.1 Services

2.1.1 Web Services

Web services provide access to static or generated Web pages and to allow download of new software versions to the trading client. The following schema shows these components.



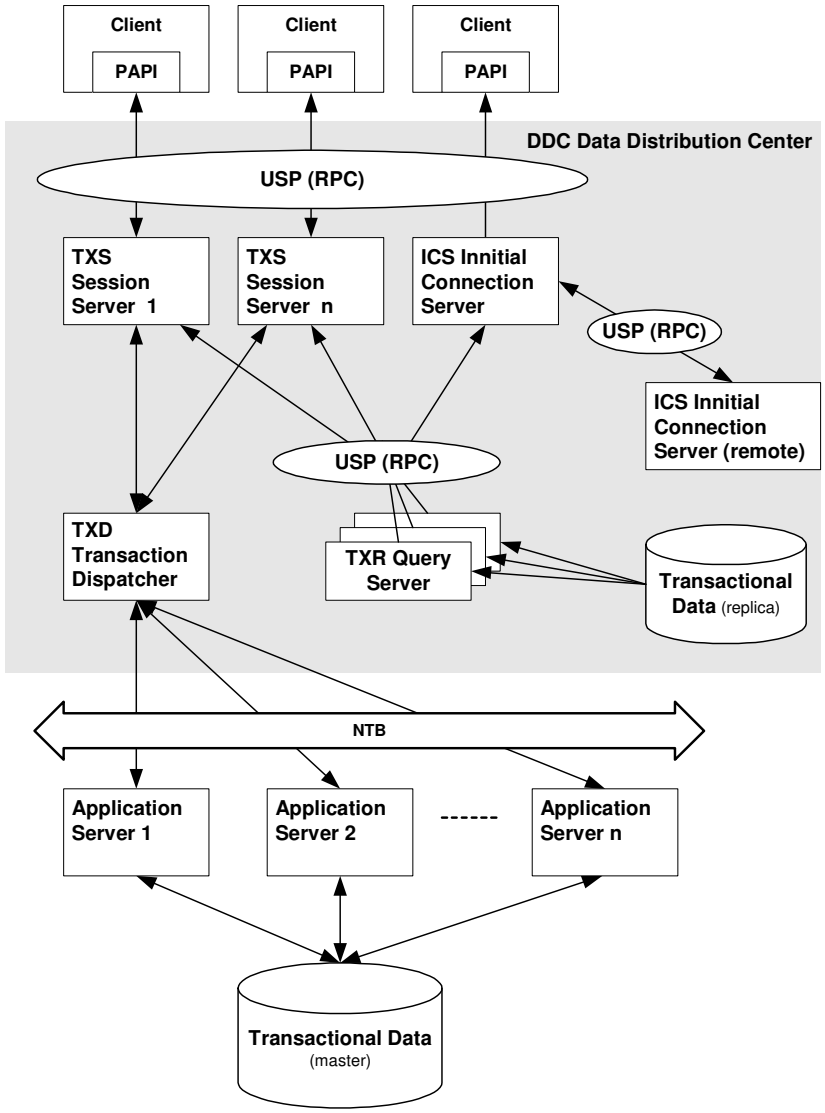
The Client is connected either directly to the DDC or to the Concentrator. On the DDC, the Web Local File Server handles HTTP requests through the WLFS Service. On the Concentrator the HTTP protocoll is redirected by the built-in Web proxy service to the Web server on DDC. TCP/IP Cluster alias is used to spread the load among the available nodes.

2.1.2 Synchronous Session Services

In synchronous mode the client sends a request message to the session server and waits until this server sends a response message back. The client request is blocking, the client cannot perform any other action in the mean time. The origin of the activity is always on the client site.

The synchronous exchange of messages is possible only within a context of a session established with a connection to a service. The duration of the session is not limited in time and is only terminated with the disconnect request from the client. The management of sessions between the clients and servers is a native function of the USP RPC subsystem.

The schema below shows the synchronous communication.



Several types of Session Services exists on a each DDC Node:

Initial connection Server ICS

Remote Initial connection Server ICS-REM

Transaction and Session Server TXS

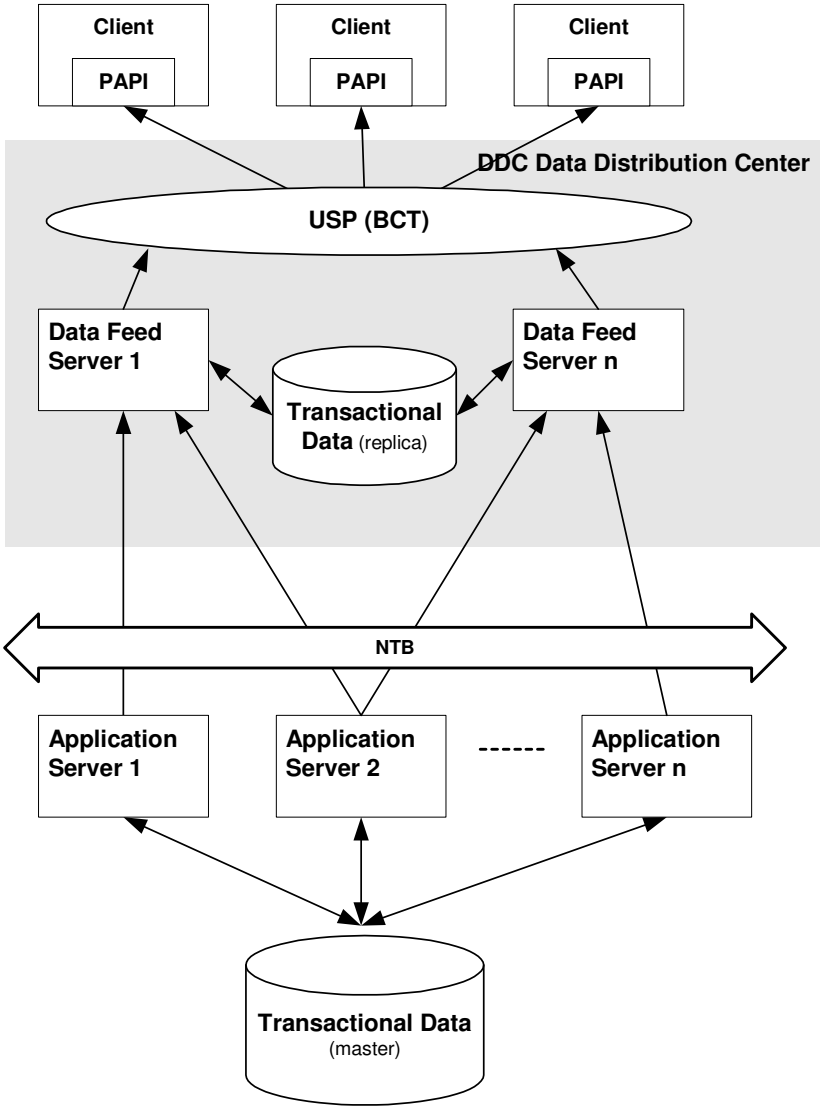
Query Server TXR

2.1.3 Asynchronous Broadcast Services

Asynchronous communication is a used to notify the client about actions performed on the server. The origin of the activity is always on the server site. Client systems must subscribe to a service in order to receive the desired information. The receival is asynchronous e.g. the client is not blocked by the receival of the data. It usually starts an extra tread to handle the incomming data stream.

The dissemination of broadcast messages to all subscribed Clients is a native function of the USP BCT subsystem.

Broadcast messages are originally released by the application servers running on the host, transported via NTB to the DDC nodes, processed by the Data Feed Servers and then broadcasted to the interested clients.



A typical flow of a brodacast message has following stages:

1. The CUI application sends a message to a business application. I.e. a trader suspension message.
2. The business application updates the host database (master)
3. A broadcast message is disseminated via NTB to the Data Feed Servers on all DDCs.
4. On a particular DDC the message is stored in the local database (replica), formatted into a client compliant format and forwarded to the USP broadcast service.
5. The USP broadcasts subsystem checks the client subscription and sends the message to all subscribed clients.

The USP Broadcast Services are organised according to markets. One central service exists for non-market information. The client subscription to a service allows additional specification of interest on a particular message type or business case. This concept provides a fine granularity of subscriptions and reduces the network traffic. The subscription is also a part of the security concept as it prevents clients to see information desired for others.

2.2 Concept of Named Storage and Container

The amount of data exchanged between the client and the server or between TXS Session Server and the TXR Query server can be quite large. The client can request a processing of a query resulting in thousands of data rows. On the other hand the client can send a large

amount of data to be processed in a single transaction. To support these requirements and to optimise the network performance, the concept of named storage has been implemented on the DDC.

The named storage is divided into storage areas. A storage area represents a dynamic storage of data and is referenced with a unique name. Primarily the client controls storage areas, but the transaction and session server and the data feed servers uses this concept for internal purposes as well. Storage area is organised in records. Record is a variable stream of bytes with length between 0 and 32k. Number of records is not limited but should generally not exceed 10k. The number of storage areas is not limited but should generally not exceed 100.

The client has access to the named storage through a defined interface. These methods allows to:

- Add data to the storage area
- Set data into the storage area (random access)
- Get data from the storage area (random access)
- Get size of the storage area
- Clear the storage area
- Process data stored in a storage area as transactional messages, replace the content with the process results.

The storage area is automatically created when it is refferenced for the first time. All storage areas are automatically deleted when the connection to the client is terminated. The concept of implicite constructor and descructor has been implemented in analogy with java and was thought to reduce the network traffic.

The data exchanged between the client and the server is passed in blocks called "containers". Container can hold multiple messages of one storage area. Containers have been introduced to optimise network traffic. The maximum size of the container is negotiated during the session between the client and the session server. The container used by the data feed servers is of fixed size. The default is 10 Kbyte.

2.3 Concept of condition handling for clients

The client is not able to translate condition code into readable text for two reasons:

The transalation table can no be stored on the client (no access to local files)

The translated text can contain variable part (names, Ids or others) not known on the client.

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.

2.3.1 Infrastructure Related Conditions

The technical diagnostic code indicates the outcome of the transaction whether the operation physically could be completed or not. The communication layer of the Client applicartion evaluates and handles this diagnostic code. The textual representation of the error is dynamically created by the server and is passed to the client as additional text parameter. The numerical values of the code are hardcoded in the Client Application. Three types of diagnostic code exist:

Success
Everything went o.k. the content of the regular message is valid.

Information
The action ends in an expected state, the content of the regular message is valid.

Failure
there was an error, the content of the regular message is not valid.

2.3.2 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 or in the related broadcast

message. The PAPI or GUI layers evaluate and handle this validation code. The textual meaning of the code is hardcoded in the Client Application.

2.4 Concept of 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)

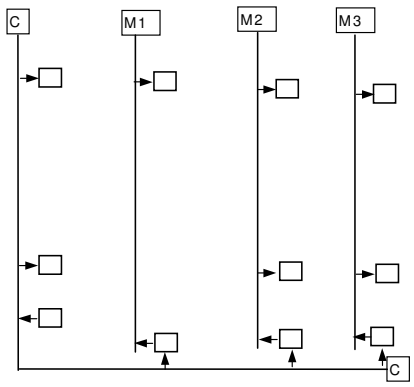
Shared RTR facilities have been introduced to allow a single application instance, i.e. the Transaction Dispatcher (TXD) to address different markets.

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 product including a market specific RTR filter mask will be implemented. This is achieved by calling the NAM facility that has been designed to create environment specific resource names.

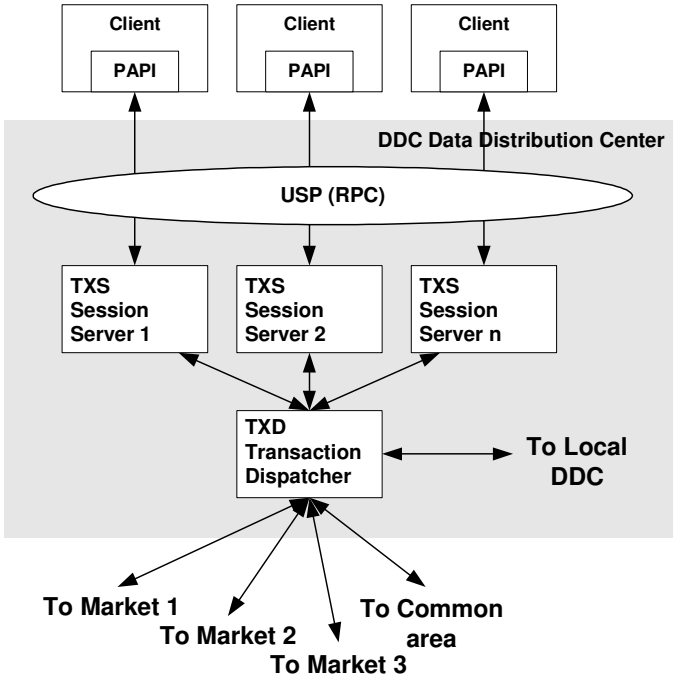
2.4.1 Market Relay Servers

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 server channel in one market and send the received messages on the Store/Forward client channel of a different market. To use concurrently NTB channels from different markets is possible because the product type and the market short name are configuration parameters to the NTB channel.



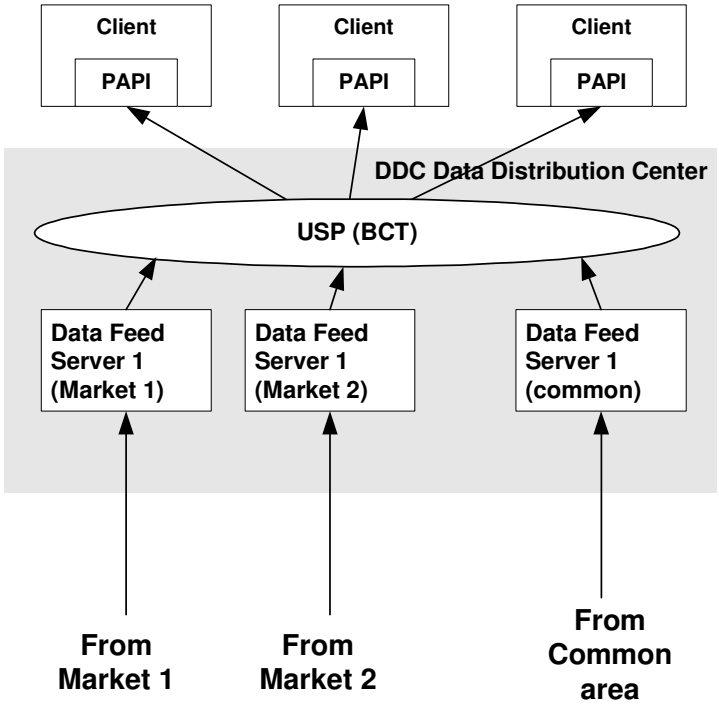
2.4.2 Routing of Synchronous Messages

The single process Transaction Dispatcher (TXD) is responsible for routing of transactional messages to the appropriate market and to the local DDC. The market information is a part of the message header and is treated as the routing key.



2.4.3 Routing of Asynchronous Messages

There is a set of Data Feed Servers for each Market and one Server for the non-market (common market) information on each DDC. The USP Broadcast Services are organised according to markets. One central service exists for non-market information. The client subscription to a certain service allows specification of interest in a particular market.

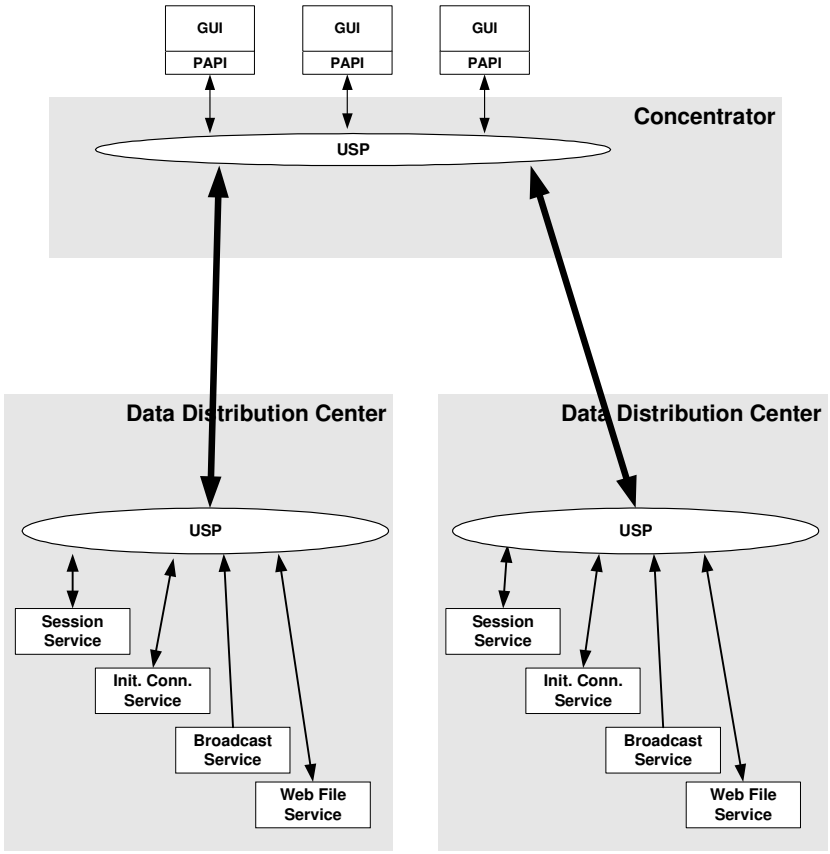


2.5 USP Concentrator

The USP Concentrator is used for customers with more then 3 concurrent users. The main purpose of this component is to distribute (fan-out) broadcast messages to connected clients. Synchronous messages are bundled and transferred between the USPs. For Web Services the concentrator plays the role of a proxy server. It does not contain any business logic.

The concentrator is running on a PC with NT4.0, Windows2000 or Windows XP located on the participant site. For security reasons one concentrator can serve users of one participant only. As the number of user increase, multiple concentrators can be installed. The concentrator requires only minimal maintenance and can be placed as unattended equipment between two firewalls. In this way security can be increased.

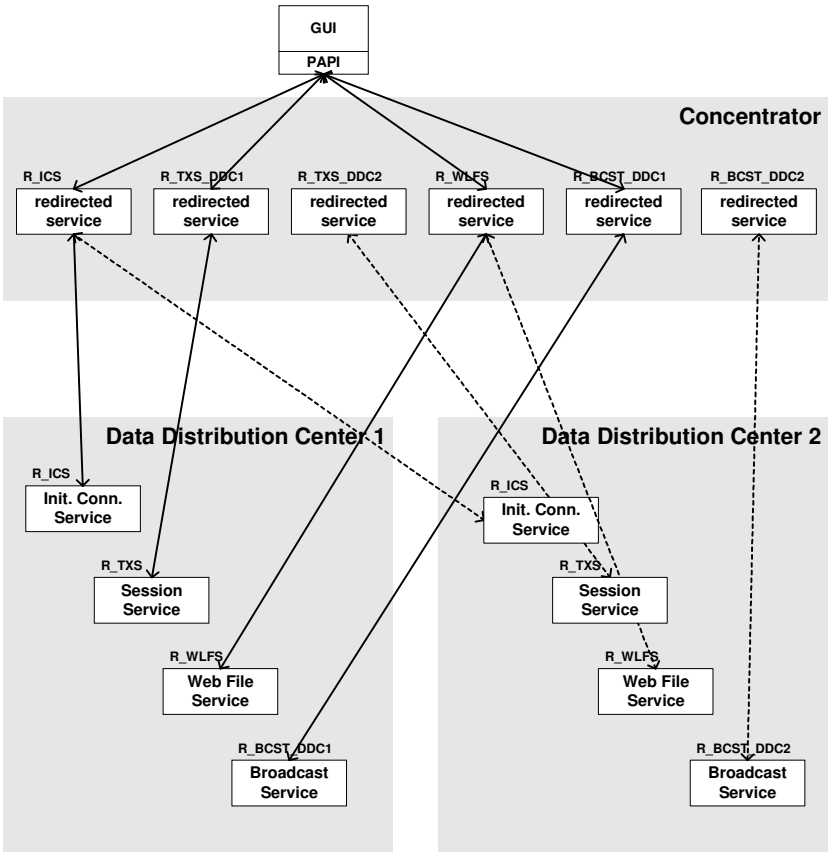
The concentrator establishes a connection through the AltaVista Tunnel Client to all required DDC nodes. Therfore each combination of DDC nodes and services must be configured to provide cross node services.



The configuration of the concentrator software is stored in the NT-registry. All concentrators are configured equally. The configuration must be changed in the following cases:

- New DDC nodes are added to the REPO system
- The names of the USP services are changed or new services are introduced.

The folowing picture shows the service redirection:

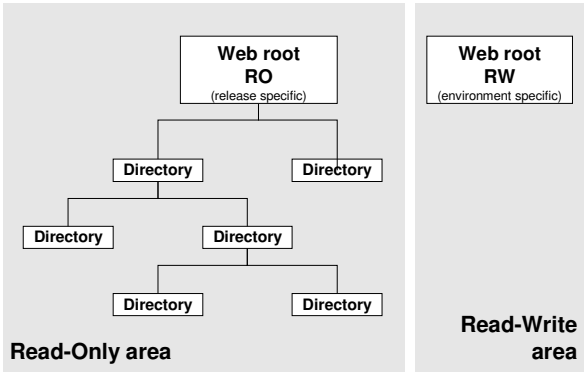


The redirection of the service R_ICS and R_WLFS is made to the cluster alias of the main DDC cluster. There is no postfix for these two services. The services R_TXS and R_BCST, R_xx_BCST do have prefix and are redirected to the particular DDC node. When the Repo Client connects, the Initial Conncention Server ICS will dispatch the request and pass back information to which service the client should connect. The concentrator is taken into consideration as the ICS knows what type of access point type is currently in use. In case of a direct connection the service names remain the same and the Repo client is advised to connect to the specific DDC node. In case of a concentrator, some service names must be extended with the node postfix but the Repo client connects to the concentrator it self.

3. Component Description

3.1 Local Web File Service WLFS

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.



Optionally the server may delete temporary files with a specific file name pattern (e.g. all files matching the pattern %%%_TEMP_*.HTML) after successful send.

The server is running on all main DDC nodes. It serves one specific environment on the particular node. The service name in the URL defines the environment. The third subdirectory in the URL is the switch of the read-only and read write area. This is necessary because of to the different nature of the files stored in these areas. The Read-Only area is release specific. E.g. exists only once for a release and is shared among several environments. In opposite to this the Read-Write area is environment specific and exists for each particular environment. The examples of valid URLs are:

```
http://146.109.231.150/bin/I06_R_WLFS/RO/struc/test/mypage.html
```

```
http://146.109.231.150/bin/I06_R_WLFS/RW/temp_page.html
```

In the example above the URLs map to these files:

```
VSI_WEBBIN:[struc.test]mypage.html
```

```
VSI_WEBPRM:temp_page.html
```

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

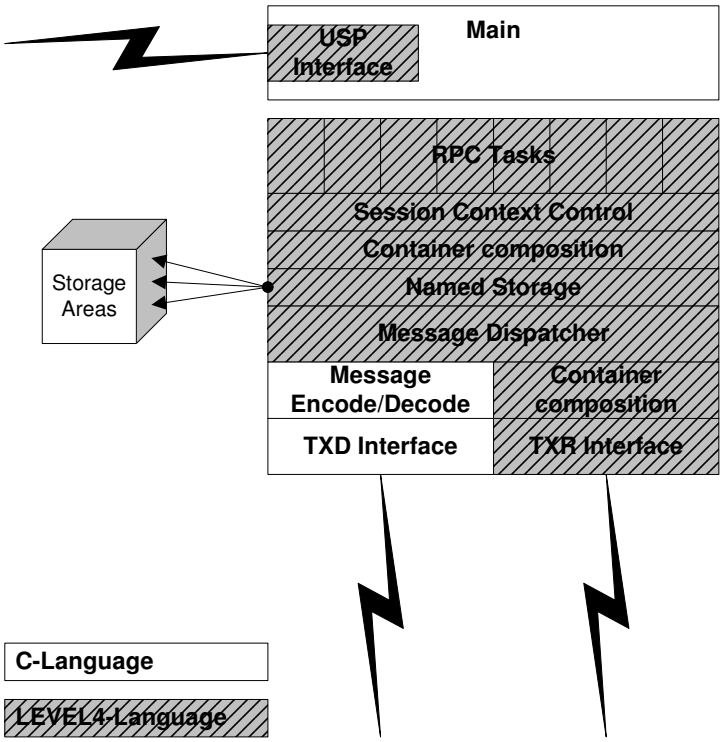
3.2 Initial Connection Service ICS

The concept allows connection between any DDC and any client. For several reasons (described later in this document) all clients of the same participant should connect to the same DDC. The purpose of the initial connection server is to find the best suitable DDC for the client, to verify that the client connects from a registered site and to check the client software version against the compatibility list maintained on the server. The initial connection server verifies the TCP/IP address of the client against the database and rejects the connection of the IP address is unknown. This TCP/IP Address (or the pseudo address of the AVTC) serves as a key to find the preferred DDC for this client in theCBCD database. For remotte DDC nodes, the client checks the availability of the node by creating a session with the ICS service on that node. If a node is not available, the server will check availability of other DDC nodes until it finds one. The server then returns information, which is required to establish a durable session context, to the client. The client terminates the session to the initial connection server immediately after.

The initial connection server is running within the USP as a stateless server on each DDC. Multiple instances are pre-started to support concurrent client connections.

3.3 Transaction and Session Server TXS

The Transaction & Session Server serve all synchronous requests from the client and passes responses back. The schema below shows the major components and their layering.



The RPC Tasks represents the granular API for the client. The creation and deletion of the session is handled in the session context control layer. The context holds information about the state of the session, about the connected client and his privileges. The data exchange is based on containers (set of concatenated messages. This allows sending and receiving multiple messages at once. Messages are passed from the client to the Transaction and Session Server TXS and processed within a transaction. The named storage described in chapter 2.2 provides an object oriented interface for messages of any type. Messages in the storage areas are converted and passed either to the TXD interface or to the business logic modules for further processing.

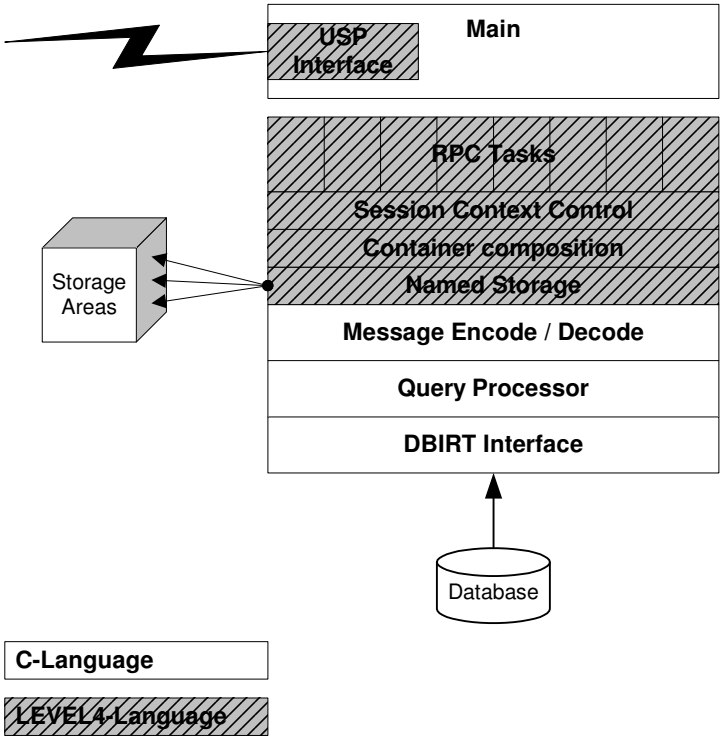
A typical client request/reply message is processed the following way:

1. A request is validated within the Transaction and Session Server. The Client must be already authenticated, logged-in and must have the privilege to process this message.
2. Dependent on the message type, the request is either processed locally (query) or passed via TCP/IP to the Transaction Dispatcher TXD. Local requests are passed to the TXR query server, which in turn retrieves the required data form one of the connected databases. The TXR query server allows fast switching between the databases without the necessity to have an open channel to all databases for each instances of the Server. Without the TXR Server there would be 200 x 5 = 1000 concurrent database connections. (200 Instances, 5 Databases). Retrieved data is passed back to the Client without conversion.
3. According to the market code the message is sent by the Transaction Dispatcher TXD process via NTB to the related business server process by using a market specific NTB client channel.
4. The business process responds with a message, which is passed back to the client over the established connection.

Optionally the client might send several request messages in sequence and business process may respond with a single or with multiple reply messages, depending on the business case. The message type indicator in the header defines the protocol.

3.4 Query Server TXR

The Qery server serve all query requests from the TXS Session Server and the ICS Initial connection Server and passes all retrieved data back. The schema below shows the major components and their layering.



The RPC Tasks represents the granular API for the client. The creation and deletion of the session is handled in the session context control layer. The context holds information about the state of the session. The data exchange is based on containers (set sof concatenated messages. This allows sending and receiving multiple messages at once. Messages are passed from the Transaction and Session Server TXS or Initial Connection Server ICS and processed within a read-only transaction. Retrieved data is encoded and passed back to the clients. Each TXR server is operatin on a single database.

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

1. A request is decoded and passed to the query processor.
2. Dependent on the message type, the request is processed and the result is retrieved from the database.
3. The result is encoded, packed into containers and passed back to the client.

The ICS is communicating with TXR with and without containers. For simplicity it uses plain ASCII data with binary delimiters to exchange messages.

3.5 Transaction Dispatcher TXD

The Transaction Dispatcher process operates as a transactional message dispatcher that reads TCP/IP messages from the the Transaction & Session Server process which subsequently are routed via NTB to either a business application on the host or to a data retrieval process residing on the local DDC. Optionally this process may return reply message(s) to the originator of a transaction.

The TXD process is installed as a single instance on a DDC supporting all active markets.

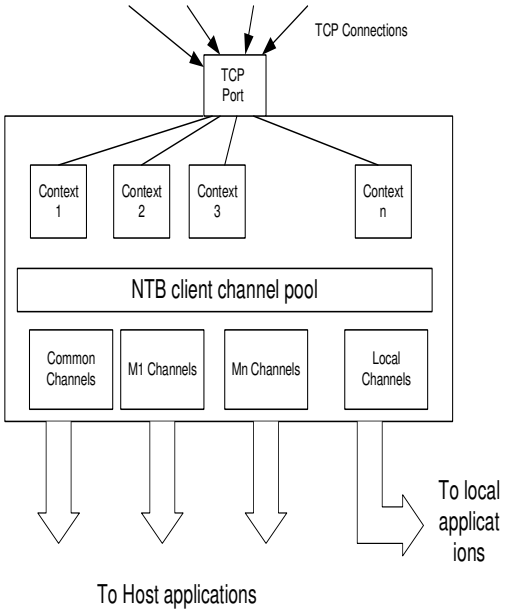
Supported message protocols are:

- single request, single reply
- single request, multi replies
- multi requests, single reply

Multi requests, multi replies are not supported

The process allocates a configurable number of NTB session client channels for :

- common area
- market 1 .. n
- local sysetem



3.6 Common Feed Manager

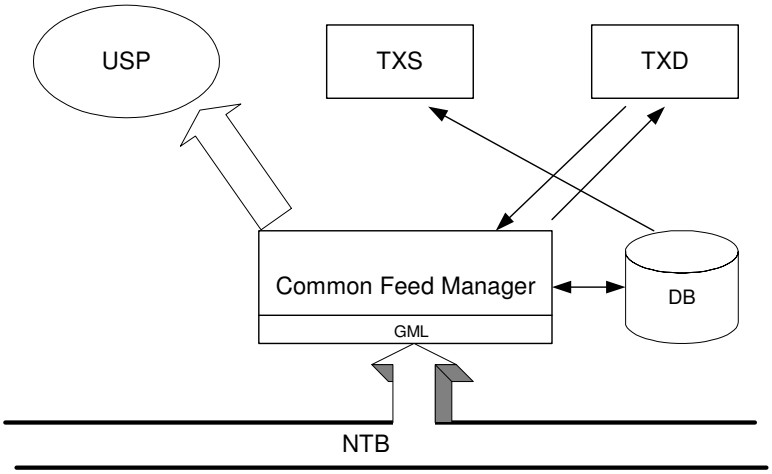
The Common Feed Manager is a standard GML application processing the Store/Forward messages released by the business applications in the common area.

One of the main-tasks of this process is to download static information stored in the CBC database upon the reception of the start of business day message that is market specific. Downloading is done by contacting the CDS server process on the host, which has direct access to the CBC database.

Some of the update messages are disseminated directly to the connetced clients via the USP services. Examples for this are the text messages.

The Level 4 layer within the TXS process handles queries supplied by the clients to get static information. A direct access to the database in the common area is done.

To be ready to satisfy initial request messages released by the client applications, this process is also configured as a session server application the DDC local RTR facility.



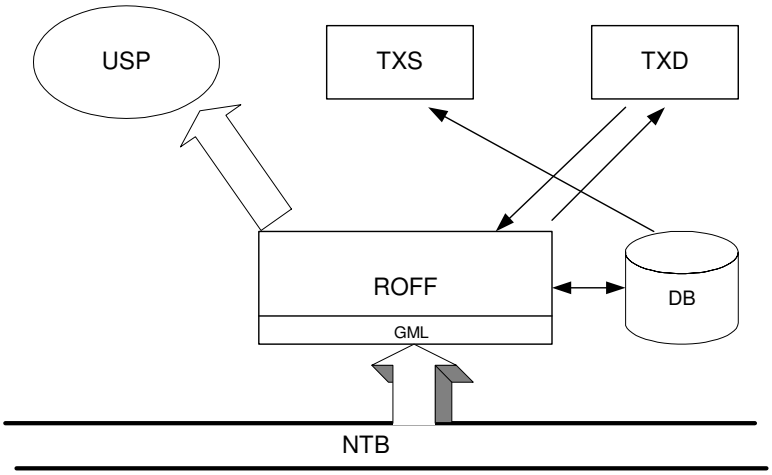
3.7 Off Market Feed Manager

The Off Market Feed manager process is a standard GML application that processes the market specific off-market Store/Forward messages and services requests for initial information of market specific data.

Whenever appropriate the Off Market Feed Manager forwards received Store/Forward messages to the connected client via so-called object messages.

Clinet requests for initial conditions are fed via the Transaction Dispatcher Process TXD. These are transactional messages and thus the ROFF process needs to operate as a NTB session server process on the DDC local RTR facility.

Client queries for market specific information are handled by the TXS process, which performs a direct access on the market specific database.



4. Sub-system External Interface Mapping

The DDC communicates with two systems:

The client.
The communication is a mix of standard HTTP protocol, USP RPC and USP Broadcast protocols.

The Backend Host.
The communication is via NTB messages.

See Use-Case documentation for the list of used and processed messages and theit content.

5. Sub-system Internal Interface Mapping

5.1 Process Shared Global Section

The purpose of this global section (shared memory) is to share information about connected clients among the processes running on the same DDC. The Data Feed Servers must know the boradcast mask of the connected clients in order to be able to sent them a message. This and other information is provided by the TXS process to the TXD process and through the global section made available to RCFM and ROFF processes.

During the participant client's authorisation phase, the following information is written into the global section:

- the participant id
- the client id
- the complete client broadcast mask
- Additional information used by the ESMAN monitor to allow monitoring and control.

The market id plus the participant id allow the Data Feed Server to figure out whether broadcast messages need to be disseminated from the current DDC. This reduces the communication to the USP, as messages for clients, which are not connected, may not be sent. This causes an optimisation of the message distribution and a performance incerease.

The existance of the global section shared by several processes makes these processes dependent on each other. The processes TXD, RCFM and ROFF must be running in order to make this global section available.

6. Database Access

Database access from processes written in C is done via the standard DBIRT facility.

The master copy of all data is maintained on the host in dthe CBC_DB and RDMK_xx_DB. Several processes replicate a subset of this information to the DDC.

The data required for the Initial Connection, Session Control, Security and Message broadcasting are stored in the CBCD Database in the following tables:

- PARTICIPANT
  Contains record for each participant. The CUI maintains this table. The following attributes are required for proper processing:
  - PARTICIPANT_ID
    Unique Identification of the participant generated by CUI as the record is created. Can not be changed.
  - PREFERRED_DDC_NODE_NAME
    Name of the DDC node to which all clients of this participant should be connected. The CUI user (OPS) will set or modify the available DDC nodes and will also set or modify this value.

  - BROADCAST_MASK
    Unique mask used to identify the participant subscription for broadcast information. The CUI will generate this value based on the PARTICIPANT_ID and NAME_SHORT. The mask can be modified only if all users of this paricipant are disconnected. (the mechanism how to ensure this is not clear yet)

- PARTICIPANT_USER
  Contains record for each participant user. The participant user with administrator rights and / or CUI maintains this table. The following attributes are required for proper processing:

  - USER_ID
    Unique identification of the user. The CUI and / or the CDM generate this ID.

  - PARTICIPANT_ID
    Foreign key to the table PARTICIPANT.

  - NAME_SHORT
    The CUI and / or the CDM create this name when new user is added. The user can modify it at any time. The value must be unique within the same participant. The user must enter the NAME_SHORT in order to perform the regular login procedure. The TXS validates the entered name against the database as a part of the login procedure.

  - PASSWORD
    Encrypted password of the user. The user maintains this field. The CUI and / or the CDM create or modify this field. The user must enter the PASSWORD in order to perform the regular login procedure. The TXS validates the entered value against the database as a part of the login procedure.

  - PASSWORD_LIFETIME
    Number of days the password is valid. The user with the administrator rights can modify the default value set by CUI when the user was added. The TXS validates the lifetime of the password and denies access for expired passwords.

  - PASSWORD_LAST_CHANGE_DATE_TIME
    The date and time when the password was changed last time. The item is updated when the user changes its password by CDM. Following rules define the validity of a password:

    PASSWORD_LAST_CHANGE_DATE = 0 => the password is not valid yet, pre-expired. The user can log-in but must immediatelly change the password value.

    PASSWORD_LAST_CHANGE_DATE + PASSWORD_LIFETIME < current_date => the password was expired and in no longer valid. The user can not log in. The Administrator must change the password In such situation.

  - BROADCAST_MASK
    Unique mask used to identify the user subscription for broadcast information. The CUI will generate this value based on the USER_ID and NAME_SHORT. The mask can be modified at any time.

  - ACCESS_PRIVILEGE
    Code which defines how the user can access the system. The possible values are: Read-write, read-only, none. The Item is maintained by the participant user with administrator rights and / or by CUI. Currently the TXS does not use this code. The privielges are controled by the tables MESSAGE_PARTICIPANT_FUNCTION, and PERTICIPANT_USER_FUNCTION.

- PARTICIPANT_USER_FUNCTION
  This table defines the roles of the user and indirectly gives the user rihts to execute only certain functions. The CUI maintains this table. The following attributes are required for proper processing:

– USER_ID
Foreign key to PARTICIPANT_USER table.

– FUNCTION_ID
Foreign key to PARTICIPANT_FUNCTION table

The TXS reads this table and makes sure that client requests match the allowed functions.

- MESSAGE_PARTICIPANT_FUNCTION
This table defines the relationship between messages and functions. The following attributes are required for proper processing:

  – MESSAGE_CODE
  Code of the message

  – FUNCTION_ID Foreign key to PARTICIPANT_FUNCTION table defining the assignment of this message to a certain function

  This table is loaded with values at the database creation. There is no interactive function available to change it. The TXS reads this table and makes sure that client requests match the allowed functions.

- ACCESS_POINT
Each access point is registered in this table. (Access Point represents the AltaVista Tunnel Client). The CUI maintains this table. The following attributes are required for proper processing on the DDC:

  – ACCESS_POINT_ID
  unique identification of the access point. The value is generated by CUI when the record is added.

  – CLIENT_PSEUDO_IP_ADDRESS
  Unique ip-address generated by the AltaVista tunnel Server and used to connect to the TXS server. This address is validated by the TXS. The CUI-user (OPS) will set the proper value. The value cannot be modified.

  – NAME_SHORT
  Unique name of the access point. This is for clarity only. The name is set by the CUI-user (OPS) and can be modified at any time.

  – ACCESS_POINT_TYPE
  This field reflects the type of the connection made from this access point. It can be Public_Internet, Direct_internet or Concentrator.

- ACCESS_POINT_CONNECTION
This table defines the relationship between participants and allowed access points. The CUI maintains this table. The following attributes are required for proper processing:

  – PARTICIPANT_ID
  Foreign key to PARTICIPANT table

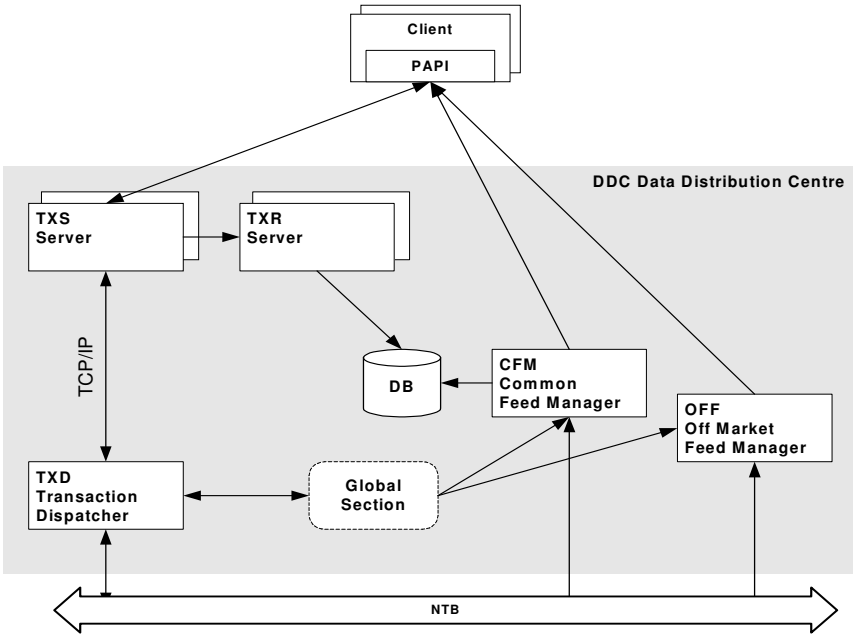
  – ACCESS_POINT_ID
  Foreign key to ACCESS_POINT table

7. Process Monitoring & Parameters

The process configuration on the DDC is split into two areas. The first area covers the configuration to process running under OMC control and the second area focus on the processes under USP control. The TXS and TXR processes are controlled by both OMC and USP and is descibed in both sections according to the related topic.

7.1 OMC Controlled Processes

This section provides an overview of the OMC controlled processes running on each DDC.



The Transaction & Context Server (TXS) process is pre-started by USP. This process cannot be started or stopped by OMC, but uses an OMC infrastructure for monitoring state and events. There may be up to 200 instances of these servers per DDC.

The Query Server (TXR) process is pre-started by USP. This process cannot be started or stopped by OMC, but uses an OMC infrastructure for monitoring state and events. There are multiple instances serving each of the 5 databases (CBC_Dx_DB RDMK_xx_DB) 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. 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.

The COMMON_FEED_MANAGER processes the feed from the operations applied to the Common Business Control (CBC) database. A sub-set of the messages is 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

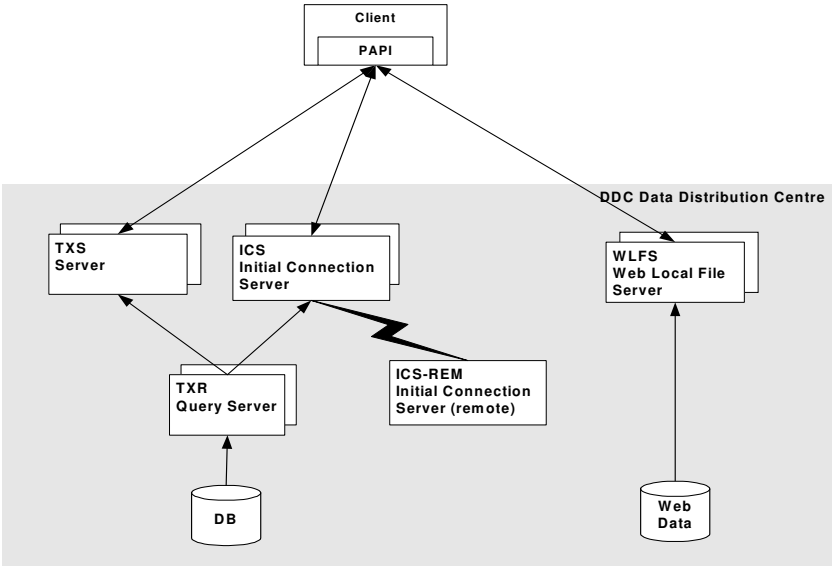
from a particular participant. There might be a need to also store the global limit values – to be decided.

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.

7.2 USP Controlled Processes

This section provides an overview of USP controlled process configuration.



Configuration of all USP-controlled processes is stored in the binary configuration file USP$DIRECTORY:USP$CONFIGURATION.CFG. The USP Monitor utility provides commnds to manipulate the configuration data. To make the configuration simpler, a central procedure has been written. This is VSI_BIN:USP_CONFIG.COM. To support several environments with different types of configuration, template files exists for each environment type. Following templates are supported:

USP_CONFIG_Txx.COM – for test environement

USP_CONFIG_Ixx.COM – for integration environement

USP_CONFIG_Sxx.COM – for system test environement

USP_CONFIG_Mxx.COM – for member test environement

USP_CONFIG_Pxx.COM – for production environement

Within these files all required parameters are defined as DCL symbols. The configuration procedure uses these symbols to generate the USP configuration. See the description of the procedure for passed parameters and details.

Monitoring and control of USP-controlled processes is implemented within the command procedure VSI_BIN:USP_CONTROL.COM. This procedure takes following parameters:

P1: * for all or service name (TXS, ICS, etc.)
P2 – P8 command
Command can be: SHOW, START, STOP, ENABLE, DISABLE.

Multiple individual commands can be given at the same line, sepparated by blank or comma. They are executed in the given order. E.g. the command:

```
@VSI_BIN:USP_CONTROL.COM * DISABLE, STOP, START, ENABLE
```

will cause a re-start of the entire system.

The ENABLE command puts a running service in a state in which it accepts connections form clients. The DISABLE does the opposite.

7.2.1 USP Services

Alle USP services are configured for a specific environment. Therefor they have a environment prefix. Following example shows all possible services:

T63_RTXS	Transaction & Session Service
T63_RICS	Initial Connection Service
T63_RICS_REM	Initial Connection Service (remote)
T63_WLFS	Web Local File Service
T63_RTXR	CBCD Database service
T63_RTXR_UK	RDMK_UK Database service
T63_RTXR_IM	RDMK_IM Database service
T63_RTXR_CH	RDMK_CH Database service
T63_BCST	Common Broadcast Service
T63_BCST_UK_node	UK Market Broadcast Service
T63_BCST_IM_node	IM Market Broadcast Service
T63_BCST_CH_node	CH Market Broadcast Service

All required services and their paramaters can be created or modified with the configuration procedure mentioned above. The deletion of the services is not supported and must be done manually with the USP Monitor utility.

7.2.2 USP Broadcast mask

USP provides the ability to broadcast messages from the DDC to the connected clients. The client controles the granularity of the subscription via so-called broadcast mask. The USP delivers only messages which correspond to the broadcast mask of the client. Character "%" is interpreted as wildcard character. Let show two examples:

Client subscribes with the mask **`X-X---`**
The server sends a message with the mask **`%%%%X%`**
The mask does not match => the cleint wil not get the message.

Client subscribes with the mask **`X-X---`**
The server sends a message with the mask **`%%X%%%`**
The mask does not match => the cleint wil not get the message.

The lengt of the broadcast mask is fixed for a particular service. With a sofisticated layout of the broadcast mask, high security level can be reached and network traffic can be optimised.

Broadcast mask layout

Attribute	Format	Comment
Participant_code	X(7)	Participant is (decimal) is used to distinguish participants.
Participant_user_code	X(7)	Client id (decimal) is used to distinguish clinets.
Client instance	X(8)	The PID (hex) of the allocated server is used to distinguish client instances.
Interest Mask	X(26)	Interest mask indicating interests of the running client. Each byte represents a certain business case.

Example:

```
000134500001238FF6AD01---X-X----------------------------
```

When the client subscription looks like the mask above (0001345 is the participant Id, 0000123 is the client Id, 8FF6AD01 is the instance id and the rest is the interest mask) then the server sends a specific message to all cleints of this participant by using the following mask:

```
0001345%%%%%%%%%%%%%%%---X----------------------------
```

7.3 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), IM (International Market) or UK (United Kingdom).

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.

7.4 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_RTXD 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 RPC, port 9101 for BCT communication with all clients and port 80 for Web Services.

7.5 Access Point Configuration

The access point has been introduced to support a flexible concept of client connection configuration. Access point represents one Alta Vista Tunnel Client installation. This can be an individual PC used by a single trade or it can be a Concentrator used by multiple traders of the same participant. To support all such situations, access points are assigned to participants, not to individual users. One participant can have multiple access points and all are treated equaly. E.g. participant users may connect through any access point assigned to the participant. Even multiple connections at the same time are possible.

Multiple participants must not share the same access point. E.g. there is 1:n relationship between participant and access point.

In the test environment this would cause troubles as developers and testers would be bound to a specific access points. For this reason the test data configuration allow m:n relationship between participant and access point. This results in configuration where any user can log-in from any access point.

8. Functional mapping

N/A. There is no functional specification available.

8.1 Non-functional Requirements

In order to be able to install new DDC systems, guidelines will be provided explaining how to copy the information from a DDC that is synchronised with the Store/Forward sender applications.

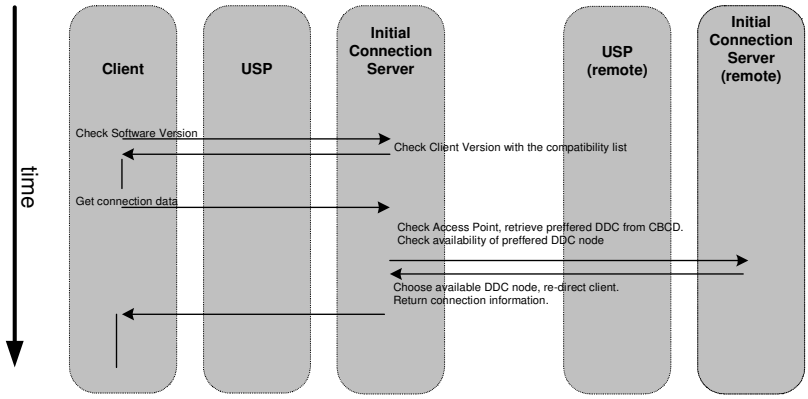
9. Sub-System Start-Up & Shutdown

9.1 Start-Up

The USP components are activated during system startup. USP controlled processes are started via the USP_CONTROL procedure. They must be stopped on each DDC individually. The remaining processes are controlled by OPS via ESMAN and started on demand by the available commands.

9.1.1 Initial Connection Sequence

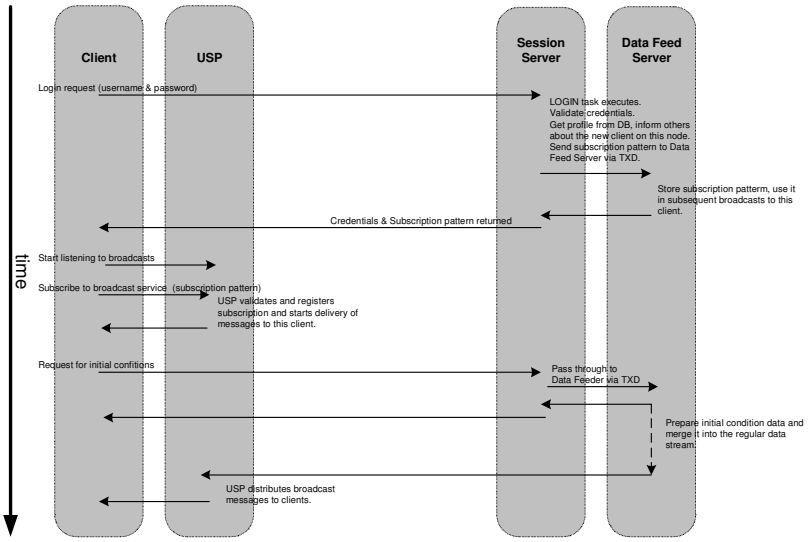
Once the Client is connected Session Start-up sequence is executed to authorise the client, load static data and establish initial business conditions. The following schema shows this sequence.



The connection is valid only if the IP-address matches the CLIENT_PSEUODO_IP_ADDR of a access point. After requested data is passed back to client, the session terminates immediatelly.

9.1.2 Login Sequence

The client must undergo several states and execute multiple steps before it can perform business functions. The following chart shows these sequence.



From now on the client is able to perform business functions.

9.2 Shutdown

USP controlled processes are stopped via the USP_CONTROL procedure. They must be stopped on each DDC individually. The remaining processes are controlled via ESMAN and stopped on demand with the available commands.

9.3 Standby Failover

The DDC's are designed to operate as standalone and independnet systems and thus do have any direct standby applications in place. Redundancy is achieved by starting and operating several DDC systems in parallel.

In case of a complete system failure the host application is not affected and the DDC will transparently recover after restarting by synchronizing the Store/Forward message queue.

The connected client applications are notified about the loss of the communication line and will try to re-connect again. They will be automatically re-directrd to one of the surviving DDC's. The order of the DDC's to connect is stored in a configurable connection list containing the preferred DDC's. Peformance on the DDC systems with more connected clients obiously will degrade to a certain degree. Planning the number of clients per DDC needs to consider the failure scenarion and leave some spare under normal conditions.

Fallback of the clients to their original DDC's is expected to occur the next day assuming the failed DDC could be recovered in meantime.

9.4 Partial Failure

In case of an unexpected process termination we presume that automatic restart is intiated by the NAC (Node Application Controller) process. Partial failures of the USP controlled processes are solved by the USP itself. The cleint re-starts the application and connects to another free server. USP keeps the number of running server instances constant.

Partial failures of prcoesses sending broadcast information via USP to the client applications (PAPI) may result in duplicate information delivered which needs to be encountered by the PAPI via the unique sequence numbers assigned to the different types of messages.

10. 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.

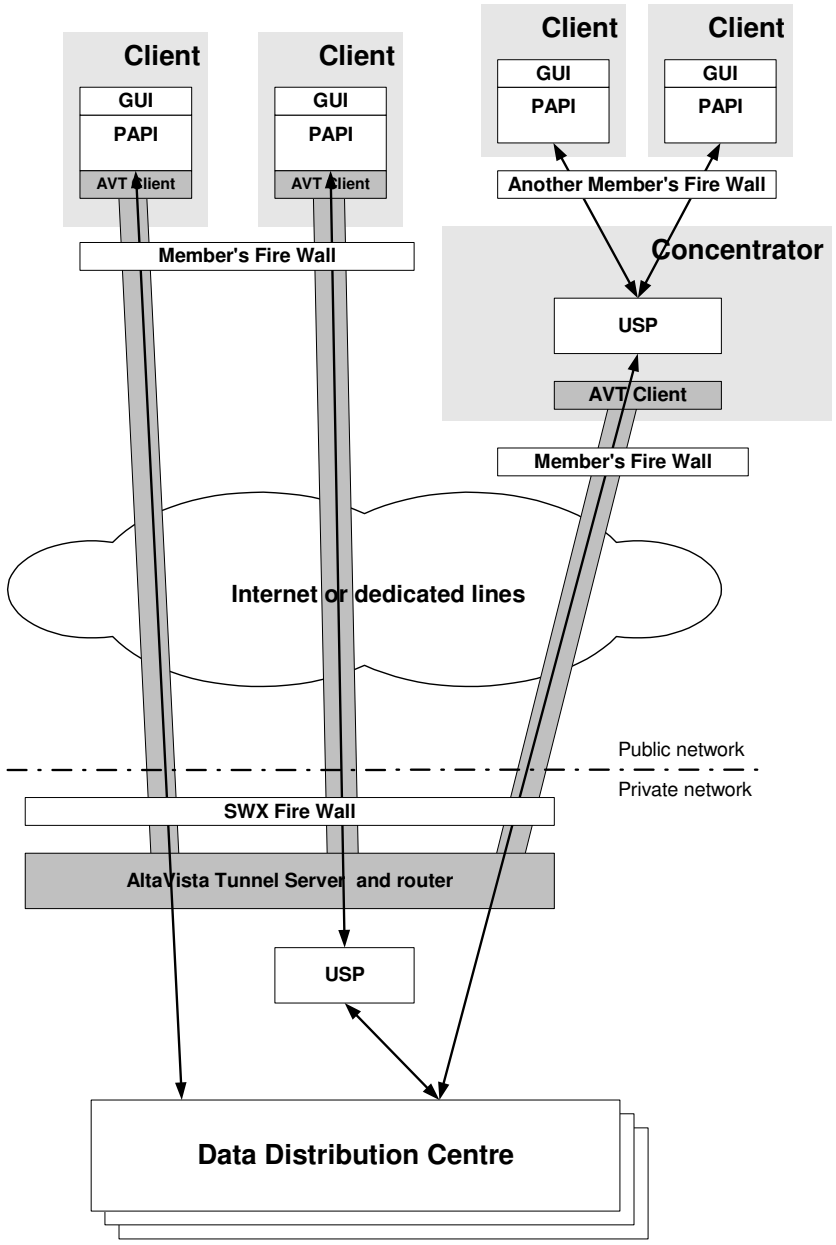
This processing is done on the DDC by the RCFM_COMMON_FEED_MANAGER process. List of tables to be downloaded can be referenced in the RCFM process specification. The specification of queries is included in the RTXS process specification.

11. Appendix 1 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 is 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.

11.1 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. The initial connection verifies this server address in the database and rejects connections with invalid IP address.

11.2 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.

11.3 Encapsulation (tunneling)

Encapsulation (tunneling) 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 cannot be subject of an attack from parties not participating in a tunnel connection. AltaVista Tunnel provides encapsulation of any TCP/IP based network protocol.

11.4 Client Authorisation

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

11.5 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. Any time the Cleint makes a request and passes information to the server, credentials stored in the server context will replace credentials passed in the request. Even when the Client would fake the message, the security is not broken, as the server will discover this kind of manipulation.

11.5.1 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. For ongoing identification the client has a short name and full name both unique within the participant.

11.5.2 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 by the transaction server.

11.6 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.

11.7 Client Privileges

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

The Client may perform only a certain functions. Functions are mapped to messages. In this way the client is allowed to send only a sub-set of messages. Both function, messages a their relationship as well as their relationship to clients are stored in the database. The transaction and session server TXS loads this data at successful login into a cache and checks them each time a message is processed. The message represents the finest granularity of access control.

11.8 USP Concentrator Configuration

The following section shows the configuraton of the USP concentrator in production environment:

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ]

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP]

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050]
"ExePath"="C:\USP Concentrator"
"ConfigPath"="C:\USP Concentrator"

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\Hosts]

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\Hosts\RPRWS]
"IpName"="146.109.231.150"
"RpcPort"="9100"
"BctPort"="9101"
"HttpPort"="80"

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\Hosts\RPRWS1]
"IpName"="146.109.231.151"
"RpcPort"="9100"
"BctPort"="9101"
"HttpPort"="80"
"EchoTimeout"=dword:2

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\Hosts\RPRWS2]
"IpName"="146.109.231.152"
"RpcPort"="9100"
"BctPort"="9101"
"HttpPort"="80"
"EchoTimeout"=dword:2

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\RPC]
"Interface"="0.0.0.0"
"Port"="9100"
"TraceLevel"=dword:4
"Enabled"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\RPC\Services]

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\RPC\Services\P01_RICS]
"Type"=dword:2
"Host"="RPRWS"
"Service"="P01_RICS"

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\RPC\Services\P01_RTXS_RPRWS1]
"Type"=dword:2
"Host"="RPRWS1"
"Service"="P01_RTXS"

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\RPC\Services\P01_RTXS_RPRWS2]
"Type"=dword:2
"Host"="RPRWS2"
"Service"="P01_RTXS"

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT]
"Interface"="0.0.0.0"
"Port"="9101"
"TraceLevel"=dword:4
"Enabled"=dword:1

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services]

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services\P01_BCST_RPRWS1]
"Type"=dword:2
"Host"="RPRWS1"
"Service"="P01_BCST_RPRWS1"
"GroupKeySize"=dword:30

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services\P01_BCST_RPRWS2]
"Type"=dword:2
"Host"="RPRWS2"
"Service"="P01_BCST_RPRWS2"
"GroupKeySize"=dword:30

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services\P01_BCST_CH_RPRWS1]
"Type"=dword:2
"Host"="RPRWS1"
"Service"=" P01_BCST_CH_RPRWS1"
"GroupKeySize"=dword:30

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services\P01_BCST_CH_RPRWS2]
"Type"=dword:2
"Host"="RPRWS2"
"Service"=" P01_BCST_CH_RPRWS2"
"GroupKeySize"=dword:30

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services\P01_BCST_IM_RPRWS1]
"Type"=dword:2
"Host"="RPRWS1"
"Service"="P01_BCST_IM_RPRWS1"
"GroupKeySize"=dword:30

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services\P01_BCST_IM_RPRWS2]
```

```
"Type"=dword:2
"Host"="RPRWS2"
"Service"="P01_BCST_IM_RPRWS2"
"GroupKeySize"=dword:30

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services\P01_BCST_UK_RPRWS1]
"Type"=dword:2
"Host"="RPRWS1"
"Service"=" P01_BCST_UK_RPRWS1"
"GroupKeySize"=dword:30

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\BCT\Services\P01_BCST_UK_RPRWS2]
"Type"=dword:2
"Host"="RPRWS2"
"Service"="P01_BCST_UK_RPRWS2"
"GroupKeySize"=dword:30

[HKEY_LOCAL_MACHINE\SOFTWARE\COMPAQ\USP\T050\HTTP]
"Enabled"=dword:1
"Interface"="0.0.0.0"
"Port"="80"
"TraceLevel"=dword:4
"Redirection"=dword:2
"RedirectionHost"="RPRWS"
"InternalServices"=dword:0
"SessionServices"=dword:0
"FileServices"=dword:0
"FileServer"="http_fileserv"
"RootDir"="C:\USP Concentrator"
```