



A Community-Source FpML Matching System

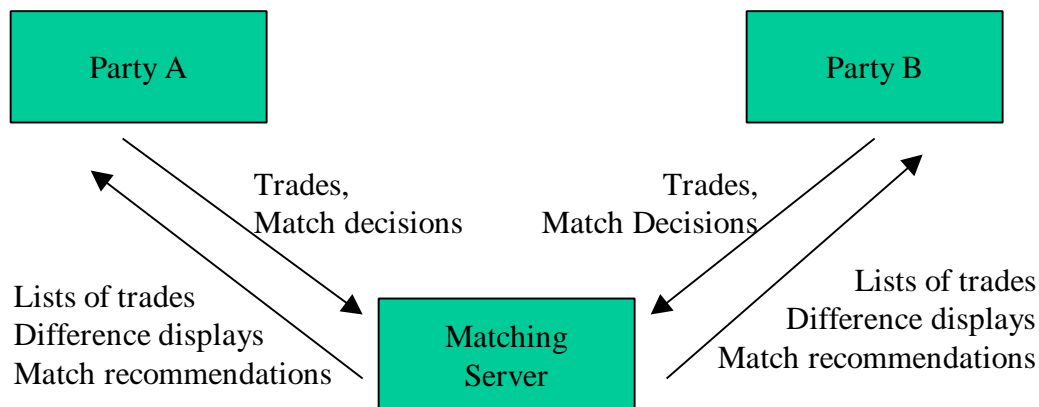
Introduction

This white paper describes how several of the Gem Soup FpML tools can be used to create a flexible, low cost trade matching system using Internet technologies. It describes the business problem, the desired solution, how Gem Soup tools would be used to implement the solution, a possible technical architecture, and various considerations.

Objective/Business Problem

- Provide same day OTC derivative trade matching for products not supported by shared trade capture services.
- Allow flexibility and rapid evolution in product representation.
- Minimize the amount of shared software and product representation that is required, to keep the service simple, effective, and easy to adapt.
- Ensure user interaction with and confirmation of matching results.

Conceptual Solution



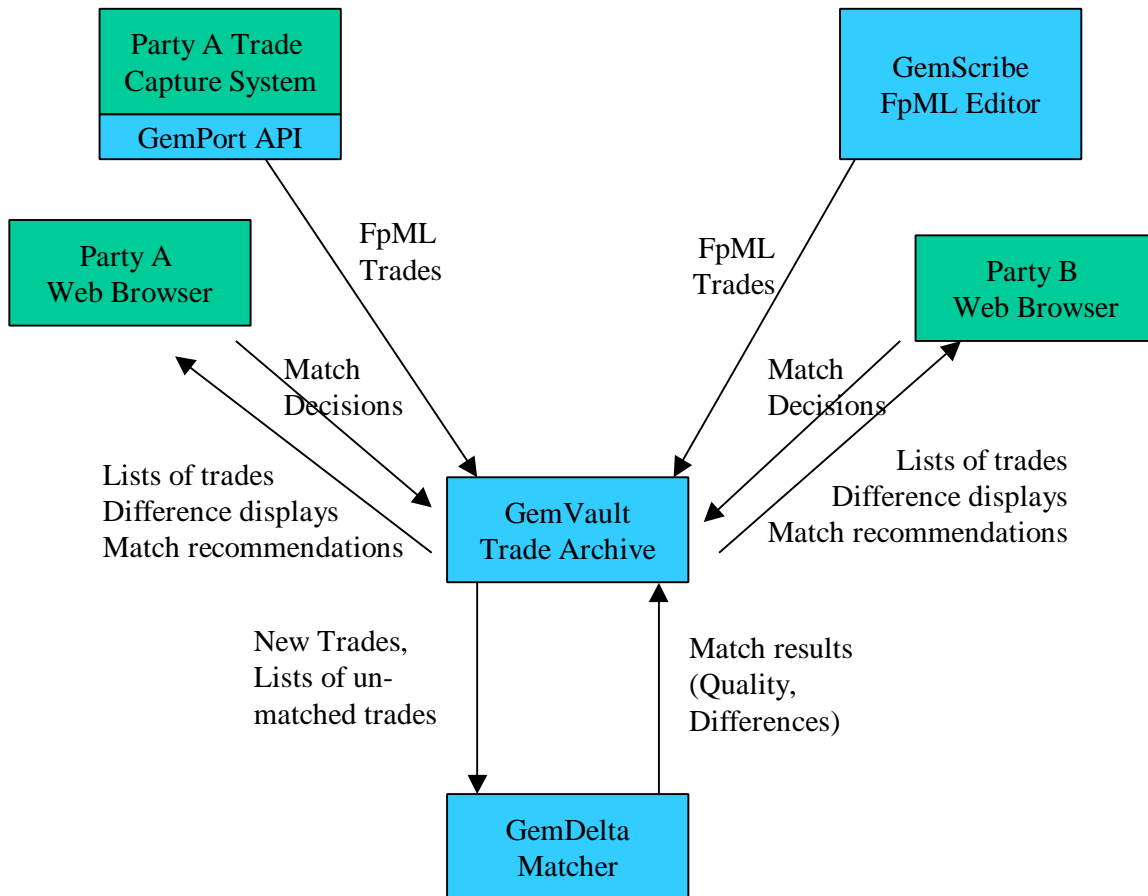
- Participants' trade capture systems output FpML representations of the trades that they enter.
- FpML is sent to a shared matching system, with an indication of the counterparty to match with.
- The matching system compares trades to corresponding trades from the other counterparty, and generates a best match and a list of differences for that match. The matching algorithm should not require much specific product knowledge, so that new products can be introduced with minimal effort.
- A user reviews the recommended matches and may accept the match recommendation
- The user reviews the list of differences between the matched trades and may modify the trade if necessary to correct inconsistencies.
- Each time the trade is modified, the service will compare the trade to the corresponding trade (if the match has been accepted) or to all trades the counterparty has submitted.
- Once the differences have been eliminated or explained, the user may indicate that the trade is "complete", i.e. no more matching is required. The matching information is then available for downstream processing if desired.



- Note that once there is a 100% match between trades, the matching system theoretically could automatically mark trades as complete. However, experience shows that human confirmation is usually desirable to prevent accidental false matches when multiple trades are nearly identical.

Software Architecture

Matching System Software Architecture



Client Side – trade submission

- Participants with existing trade capture systems can use the GemPort Programming Interface for FpML submission. This API provides a very simple programming interface with a configurable product template and output method code generation to allow rapid development of FpML outputs from existing trading systems.
- Alternatively, participants not able to easily modify their trade capture systems may use the GemScribe FpML editor to generate FpML documents. The GemScribe editor also uses a template-driven architecture (with templates very similar to those used in GemPort) for displaying and editing FpML documents in Internet Explorer.



Service Side

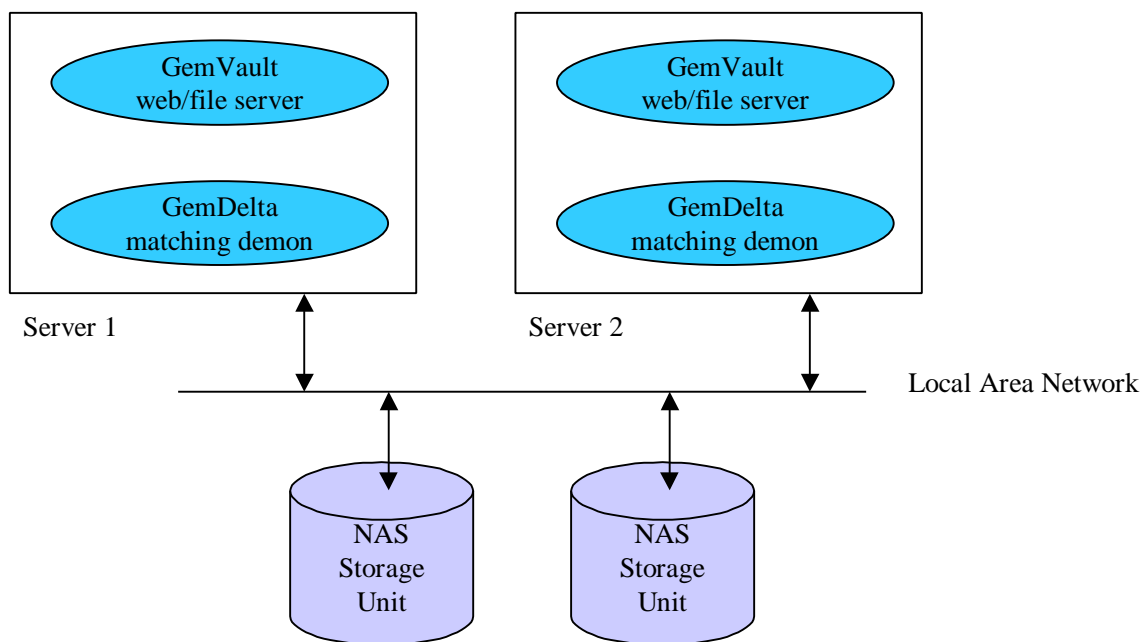
- The GemVault trade archive is used to hold and organize FpML trades. This web server based archive allows participants to send FpML files to the matching system and organize them logically to facilitate matching, and to record accepted matches. It also allows participants to display the trades they have submitted and the match results.
- The GemDelta FpML matcher is used to incrementally compare trades in the GemVault, each time a new trade is added.
- A series of views in GemVault allow users to see which trades, versions of trades, and matches are currently in the archive, and to record which matches are accepted.

Client side – matching workflow

- To display and evaluate matches, participants may use web browsers to display trade lists, match quality, etc.
- The GemDelta difference viewer allows detailed trade differences to be displayed using a web browser.

Technology Architecture – Matching server

Matching Server Technical Architecture



Hardware

For the central matching system itself, for a shared, multi-participant system we recommend the following hardware configuration:



- Redundant dual processor Intel Pentium 4 or Xeon Linux (e.g. Red Hat Linux 7.2) servers running Apache web server, GemVault, and GemDelta
- Redundant Network Attached Storage (NAS) servers running at RAID 5 for fast, convenient, and reliable storage.

The Linux hardware is reliable, flexible, excellent for networking and security, and cost-effective. We have found that Sun's Java performance may be somewhat slower on Linux than on the same hardware running NT, but we believe that Linux is a more appropriate server environment, and offers excellent cost-benefit tradeoffs. (Alternatively commercial Unix servers (e.g. from Sun) may be used, at a somewhat higher cost.) The NAS storage provides excellent performance and simple installation, and allows a very scalable installation as the system grows.

Colocation/Networking

We recommend installing the system as follows:

- Option 1: Hosted at a co-location center closely connected to the Internet backbone. A number of vendors provide co-location (rackspace) services, which provide space, security, power, network access, and some minimal network and system administration support. These can be located either in financial centers (for convenient access) or in remote locations (for security).
- Option 2: Hosted by and for each participant. In this model, rather than sharing a single matching server, each participant would have its own. Participants would be responsible for writing FpML trades both to their own server and to that of their counterparty. (In this model a somewhat smaller hardware configuration could be used for each server).

Backup

- NAS devices allow instant snapshots and online recovery of old snapshots, which protects against accidental data loss or damage caused by user or operator error or software problems.
- Since in a matching service most or all user data is generated new each day, there is little need for backup, other than to safely store logs for analysis and audit purposes
- Back ups can be done directly within the hosting center, or since the data volumes are relatively small, can be done offline across the Internet using secure copy utilities.

Disaster recovery:

There are several disaster recovery options, depending on how quickly service needs to be re-established in case of a disaster.

- Low cost option: dedicated Linux servers can typically be set up in <24 hours by hosting/colocation providers. It may be sufficient to establish an arrangement with a geographically separated colo provider, and define a disaster recovery process. For a recovery site a simpler technical architecture without NAS devices should be adequate for short term operation. This option would allow operational recovery in ~24-48 hours of a disaster affecting the original colo provider.
- Full Disaster recovery: Duplicate part or all of the infrastructure at a second site. (At a disaster recovery site, there is less need for redundant hardware, so it may not be necessary to duplicate all the hardware.). This should allow disaster recovery within a few hours.

Scalability

- Based on current benchmarking, the described technical architecture should be adequate to support matching of in the order of several thousand trades per day (including several versions of each trade). (This capacity will vary depending product complexity, matching parameters, and participants' behavior.)
- If increased trade volume requires higher amounts of CPU resource (to maintain matching and browsing performance), this architecture can easily be scaled by the addition of extra Linux servers to perform the matching processing.



Security

- Message transfer traffic should be encrypted using 128 bit SSL, Apache 1.3, OpenSSL, mod_ssl (or equivalently Stronghold from RedHat).
- User authentication could be done using username/password, e.g. using HTTP Basic Authentication.
- For better security, here is a recommended authentication mechanism:
 - Client side X.509 certificates can be privately generated by the matching service and downloaded using SSL to the authorized client machines, and installed in their browsers/attached to their applications.
 - Downloads of the client certificates can be controlled, e.g. by requiring e-mail authorization by the designated client side contact or by the matching service help desk, and by logging the distribution of new certificates.
 - Client-side certificate checking should then be enabled in the web server, to prevent access by unauthorized machines/users.
 - This scheme provides superior protection compared to simple username/password authentication, and excellent auditability of authorizations.

Support

- To cost effectively provide 24-hour day monitoring and application support, we recommend using a Management Services Provider specializing in web applications and knowledgeable in the financial industry.
- There are a growing number of MSPs providing a variety of services, including for example,
 - Website availability monitoring
 - System Problem detection and resolution
 - Capacity planning and on-going monitoring
 - Regularly scheduled maintenance (e.g. renewing SSL certificates)
 - Help desk functions
 - Application support

Legal issues

- Due to the potential liability risks of this type of service, it will be necessary to carefully define legal agreements.

Costs

Here is a partial list of the costs expected to set up this kind of service.

Software:

- Linux, Apache, OpenSSL, and mod_ssl are OpenSource. They are generally freely downloaded. Packaged, supported versions (e.g. Stronghold) are typically available for <\$1000/server.
- Contact Gem Soup for support costs for GemSoup-supported products.

Hardware

- The aggregate cost of two 2-CPU Intel Linux servers and two 120 GB Network Attached Storage devices should be under \$20K USD, based on recent Dell list prices.
- Colocation: Colocation (rack space and ISP) charges for the above configuration are likely to be under \$1K USD/month, assuming moderate utilization, and moderate amounts of backup storage requirements, based on recent price quotes from several American colocation providers.



Support and Management

- The largest component of the operational costs for this type of service will be support costs. A Management Services Provider is likely to charge \$10K-\$20K/month to provide 24x7 support and basic help desk services for a setup supporting a small number of clients. (This number will be affected by the number of clients, the number of support calls, the number of software changes, etc.)
- There will also be management fees for someone to take responsibility for setting up and running the services, handling issues, controlling software upgrades, etc. These fees will depend on the manager but are likely to be similar to those of the MSP for a small installation.

Miscellaneous

- There will be costs for a variety of minor miscellaneous activities, such as purchasing digital certificates, utility software, etc.

Legal and Insurance

- Legal and insurance costs for this type of service are unknown but could be substantial, depending on the nature of the agreements.

Overall

- For a small installation supporting 3-5 firms, the aggregate service costs are likely to be in the range of \$10K-\$20K/month per firm, assuming a cost effective management fee can be negotiated, in addition to initial set up costs. These costs are in addition to the Gem software support costs.

Features

Following are some of the features of the described architecture.

- **Web-browser based listings of**
 - Trades submitted by self
 - Version history of each trade
 - Match quality by trade, and match recommendation
 - Differences between trades submitted by counterparty and by self
 - Differences between trade versions
- **Simple but effective web-based workflow model**
- **Ability to integrate with existing trading systems using flexible, easy-to use, template-based GemPort APIs**
- **Customizable matching**
 - Use of matching configuration file to allow customizable match parameters.
 - Ability to incorporate or ignore client-specific attributes.

Benefits

Following are some of the benefits of this type of matching service:

- Ability for dealers to enhance interfaces to add new products or attributes. Not all dealers need to adopt new products or features simultaneously.



- Ability to for dealers to understand and customize the matching algorithms, generally through configuration files rather than code changes.
- Ability to incorporate counterparty pair-specific product or matching features.
- This architecture requires minimal changes to existing workflows and systems.
 - Simple integration with existing trade capture systems: In this architecture, existing trade capture systems output FpML and do not need to load trades from FpML (unlike with shared trade capture services).
 - This output-only approach is easier for participants to implement, particularly using the GemPort template-based APIs.
 - If desired, it would also be possible to load trades from GemVault back to participant's systems or to processing services, e.g. to include the counterparty's trade ID in the original participant's trade representation.
- Low setup cost
- Low setup complexity
- Excellent flexibility in product definitions. New products can be added by :
 - updating the GemPort templates and generating the appropriate save methods, then relinking/rebuilding the participants trade capture system
 - optionally, updating the matching configuration file (if the default matching behavior is not adequate).
 - Updating the matching viewer if necessary to view the additional information. (The matching viewer is handles most information automatically, so many changes are handled automatically).
 - Nothing else is required, since the central services does not need to validate the submitted FpML, just compare it to other FpML using a generic comparison algorithm.