XDisect use in a Market Maker scenario

This document describes the use of XDisect in a Market Maker scenario (called Component Exchange). The objective is to depict a business scenario that is easily understood and believed by a wide audience. It includes a description of the participants in the scenario, the data being exchanged by the participants and the high level flow of information. In this white paper we will illustrate XDisect's key values and primary feature benefits when applied as a fundamental component of a market maker.

Key XDisect Value: -

- Faster time to market
- Lower adoption resistance from suppliers.
- Lower adoption resistance from consumers.
- Ability to adapt to changing business conditions.
- Minimizing IT infrastructure and planning costs.

Features / Benefits: -

This white paper will highlight the benefits of building a market maker using XDisect. Some of these feature / benefits include:

- *Flexible product descriptions* - Allows more flexibility in describing vendor product offers.
- *Personalized Catalogs* - Makes generation of personalized filtered catalogs easy and efficient.
- *Multi-Vendor Catalogs* - Makes aggregation of catalogs from multiple vendors easy even when the offers are described using different taxonomies.
- *Effective query mechanisms* - Provides effective queries for products across previously incompatible taxonomies.
- *Lower Adoption Resistance* - Minimizes the adoption resistance for suppliers by allowing them to model their product data in a way that makes sense for them. Plus it makes minimal technology demands from suppliers for participation in the marketplace
- *Flexible Searches* - The ability to flexibly search and retrieve information about bids, offers, and asks. In many instances the advanced search process can fulfill the auction provisioning. This can be useful for status queries by buyers and sellers
- *Lightweight Events based interaction* –Lightweight events allow the marketplace participants to interact with each other in ways that cannot be pre-defined and need not be.
- *Fast Ramp up of participants* - XDisect makes it easier to ramp (scale) up membership in an exchange. XDisect allows the bid, offer, ask and profile information to be

maintained without a rigid schema or structure. This will be extremely useful in a marketplace where sellers and buyers all have their own backend systems that store this information in different formats. Trying to get hundreds of customers and suppliers to agree on a format can be a Herculean time consuming effort. This is compounded when suppliers are forced to suppress what they consider key data elements in order to adhere to the market schema

- *Adapts to dynamic market requirements* - XDisect makes it easy to accommodate new users (suppliers and consumers) who have special requirements. XDisect engine with its flexible data storage and retrieval also makes changing the formats of the bid offer, ask, profile documents very easy. The only place that will have to change is the front end GUI rendering logic of the marketplace. The rest of the application layers (including XDisect) do not mandate a pre-defined structure of the XML documents and hence it would be easy to change the structure without a massive re-engineering effort.
- *Allows Data to Evolve* - XDisect makes it feasible for a market maker to evolve their data infrastructure as fast as their business conditions change and as the business requirements of the market participants change

Please note that this is just a sample scenario and any resemblance to a live project is unintentional. This scenario provides only one instance of how XDisect can be used in a market maker application. Please contact chetan@pybiz.com if you have any questions about the applicability of XDisect in your problem domain

Overview

In this scenario we consider a fictitious marketplace for electronic components called ComponentExchange. ComponentExchange is hosted by E-Comp, Inc (every e-business story has to have at least a few E's right). The goal of ComponentExchange is to facilitate purchase of electronic components online for its buyers. For the sellers the advantage is that it will help them get rid of excess inventory and also give them access to customers they did not have previously. The process used will be a reverse auction between customers and interested suppliers.

Assumptions

- E-Comp, Inc has formed relationships and signed up thousands of these suppliers.
- There are hundreds of these electronic components customers wanting to do business at ComponentExchange.
- ComponentExchange makes money by charging a certain percentage per transaction completed successfully that was facilitated by it.
- A vendor must be able to supply all parts on the order to be considered viable for a given request for quote.

Basic Flow

The basic flow of the marketplace will be as follows: -

- Suppliers/Distributors wanting to sell electronic components will advertise their offerings at ComponentExchange.
- Customers interested in purchase components will be allowed to select from a catalog that is cross- vendor and is maintained by E-Comp, Inc. Optionally customers that know exactly what they want can upload a Request For Quote (RFQ) direct to the exchange totally bypassing the catalog.
- ComponentExchange will determine all the suppliers that will be viable for this RFQ(do some selection of suppliers based on pre-defined business rules of the marketplace).
- ComponentExchange will then send the RFQ or "ask" on behalf of the customer to the selected vendors.
- Vendors will reply with their bids to ComponentExchange.
- ComponentExchange will aggregate the bids for presentation.
- ComponentExchange will then present the bids to the customer
- Customer will have the option to place an order against the bid.
- If an order is placed through ComponentExchange, it will send the order to the vendor, get a confirmation & send it back to the customer as proof of the order placement.
- From here on the standard order fulfillment process takes over. We will not get into the details of that for the purposes of this exercise.

Future Features

- The component exchange will support aggregation of customer orders for similar products. In the instance of aggregation the single vendor must supply all parts will be suspended.
    - Aggregation is necessary to allow maximum customer savings by holding orders until a sufficiently high quantity can be purchased to obtain bulk rates.

Note: The market maker will generally also support a facilitator interaction model where the actual transaction does not necessarily go through the market. The buyer and seller do the actual ordering process outside the market. The market maker just gets the buyer and seller together

Scenario Background

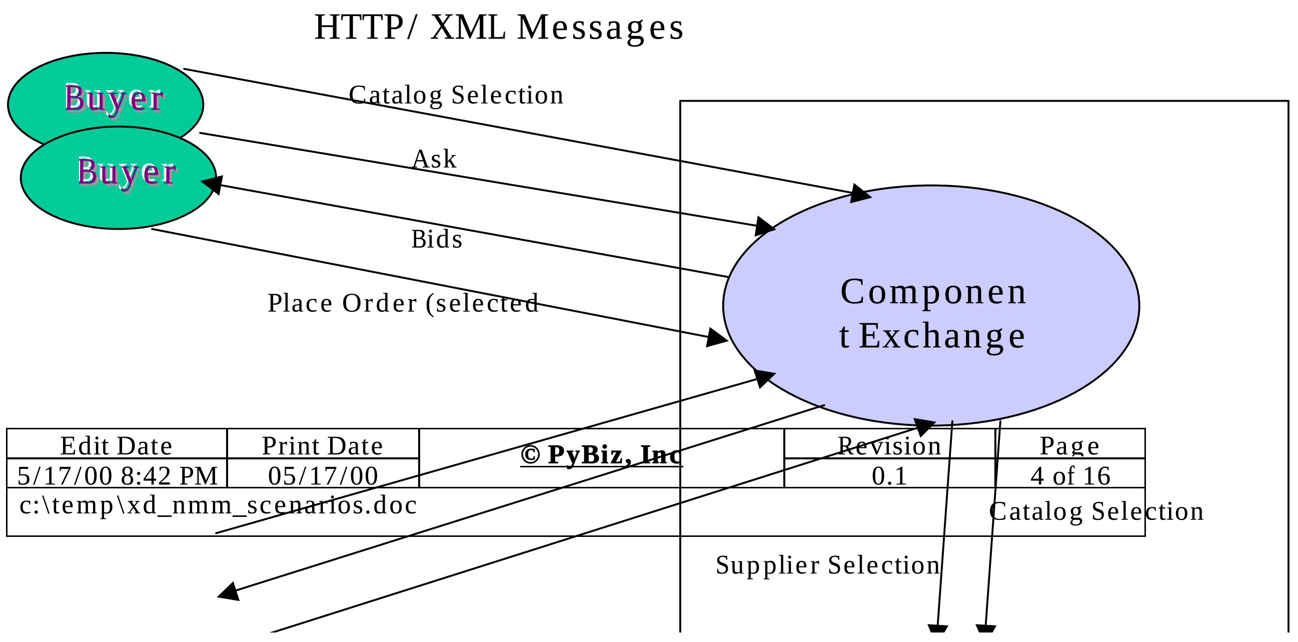
In this section we provide the basic technical background for the scenario

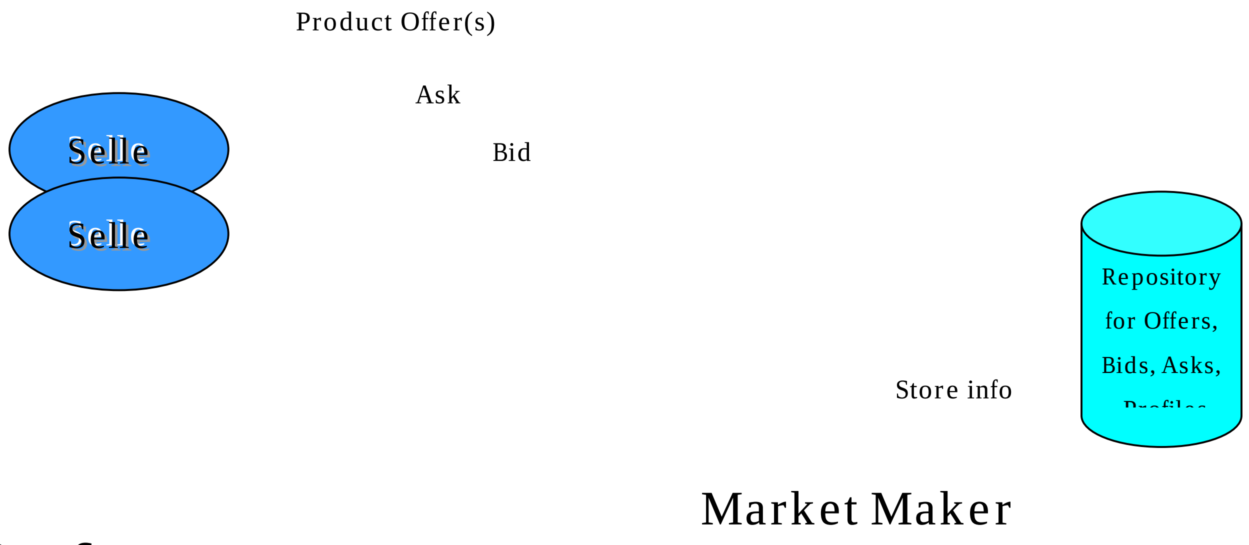
Assumptions:

1. ComponentExchange is implemented using XDisect
2. ComponentExchange has a profiling system that will maintain profiles of customers and suppliers doing business with it. Assume for simplicity that these profiles are stored in XDisect.

3. Product offers, asks & bids are also stored in XDisect.
4. All data exchanged between the various parties is in a business XML format.
5. There is no agreed upon vocabulary or schema for the components market. There is an implicit schema and a minimal set of tags. There is no rigidly defined structure to the data that all suppliers and customers need to adhere to.
6. Suppliers will share only as much information with ComponentExchange as is needed. E.g. Some suppliers may choose not to share their price and availability information while others may include color, size and application information.
7. ComponentExchange trusts the suppliers to varying levels and hence it will share varying amounts of information about customers and their requests with the suppliers.
8. Both suppliers and customers have already setup pre-negotiated contracts with ComponentExchange. Here we will not discuss the actual process of customer or supplier registration
9. ComponentExchange has all of the state of the art security in place but it is orthogonal to this scenario for now.
10. All customers, suppliers and marketplace communications are implemented using the HTTP(S) protocol. Extensions to support file based FTP interaction and emailed attachments will also be supported.
11. ComponentExchange is neutral to both buyers and sellers (hence it is neither a buy-side nor a sell-side marketplace)
12. Suppliers just need a web browser to start interacting with ComponentExchange. As they gain value from ComponentExchange they may choose to add in higher levels of support like XML/ HTTP.
13. Suppliers can interface with their legacy parts systems using their own choice of technology and are in no way bound by the choice of technology for ComponentExchange
14. Customers like Suppliers just need a web browser to interact with ComponentExchange

High Level Technical Architecture





Data Structures

Below are some samples of data structures that will be used by the market for exchanging information back and forth between the various participants

Product Offer

A supplier that wants to sell a product through ComponentExchange sends an offer. Again we emphasize that below we are just providing samples of how product offers might show up. ComponentExchange does not impose a comprehensive or strict structure on the data

```
<offer>
    <id>tr9017-offer1</id>
    <supplier>
      <name>Chetan Inc</name>
      <id>Chetan-inc</id>
      <bid_request_url>
            http://ChetanInc.com/componentexchange/cgi-bin/bidRequest.cgi
      </bid_request_url>
    </supplier>
    <product>
      <id>tr9017</id>
      <desc>general transistors</desc>
       <datasheet>http://ChetanInc.com/products/specs/tr9017.html</datasheet>
       <schematic>http://ChetanInc.com/products/specs/tr9017.html</schematic>
       <quantity>
            <max>1000000</max>
            <min_lot_size>1000</min_lot_size>
        </quantity>
        <currencies>USD</currencies>
        <currencies>Euro</currencies>
        <pricing>
            <min_price>10</min_price>
```

```
          <units>USD</units>
        </pricing>
        <delivery>
            <estimated_delivery_time>10</estimated_delivery_time>
            <units>days</units>
        </delivery>
      </product>
  </offer>
```

➢ XDisect provides the ability to store and search arbitrarily nested and complex data structures without having to pre-define any schemas, DTDs or vocabularies

Supplier/Customer Profile

Below is a sample of a supplier profile. A customer profile would be similar. NOTE: The profile here is essentially an arbitrarily nested data structure that can store whatever is relevant to the supplier/customer and to ComponentExchange

```
  <company>
        <id>chetaninc</id>
        <role>seller</role>
        <name>ChetanInc</name>
        <registered_address>
              <street>3386, Valley Forge Way</street>
              <city>San Jose</city>
              <state>CA</state>
              <zip_code>94087</zip_code>
              <telephone_number>408 733 8463</telephone_number>
        </registered_address>
        <bid_request_url>
            http://ChetanInc.com/componentexchange/cgi-bin/bidRequest.cgi
        </bid_request_url>
        <billing>
            <type>credit-card</type>
            <type>micropayment</type>
            <type>SAP direct</type>
        </billing>
        <contract>
             <discount_percent>10</discount_percent>
             <service_hours>24X7</service_hours>
             <min_turnaround_days>3</min_turnaround_days>
             <service_offered>transistors</service_offered>
             <service_offered>heat sinks</service_offered>
        </contract>
        <preferences>
             <trading_countries>Italy</trading_countries>
             <trading_countries>UK</trading_countries>
             <trading_countries>US</trading_countries>
             <currencies>lire</currencies>
             <currencies>pound</currencies>
             <currencies>dollar</currencies>
             <notify>
                 <event>bid_amount GE 100000</event>
                 <event>bid_qty GE 10000</event>
                 <receive_url>http://www.chetaninc.com/notify/receiveEvent.cgi</receive_url>
                 <receive_mechanism>asynchronous-send</receive_mechanism>
             </notify>
        </preferences>
  </company>
```

➢ XDisect provides the ability to associate lightweight events with attributes within profiles as in the company structure above.

The key things to note here are: -

- the profile may have the company's role since they could be both a seller and a buyer.
- We have not shown the security privileges for this account. These could be stored as a separate XML document related by role.
- In the preferences section, the account has specified two events . The account would like to be notified if there is any potential user that is requesting more than 10,000 units or if the request amount is more than 100,000. Perhaps the account would like to give the customer special treatment (expedited delivery, etc). Since it is a large order, the seller may want to manually verify the authenticity of the order and the customer before agreeing to fulfill it.
- Of course this profile could have been normalized further so that the preferences are stored separate, etc. But we wanted to show the relationships between the various sections of the profile and hence left it all in one profile document.
- Contracts in this case are shown to be very simplistic. ComponentExchange and its suppliers may or may not choose to store contract information in this document above.
- A customer profile would contain additional information like preferred vendor lists and approved vendor lists.

Ask/ Request for Quote

This data structure represents the RFQ generated by an interested customer. This could be either generated by the customer or by the marketplace based on the customer's navigations through their catalog GUI
NOTE: the customer may or may not want to divulge their information during an RFQ . This may be governed by ComponentExchange's rules of privacy and the customer's preferences.

```
<ask>
     <product>
       <part_num>X232</part_num>
       <line_num>1</line_num>
```

```
        <qty>100000</qty>
        <desc>90 watt varistors, 5K to 5.2 K</desc>
        <equivalent_ok>yes</equivalent_ok>
        <max_price>1000000</max_price>
      </product>
      <product>
        <type>4 ohm thermistors<type>
        <line_num>2/line_num>
        <qty>100000</qty>
        <delivery>
          <min_qty>1000</min_qty>
          <first_date>05/20/2000</first_date>
          <last_date>06/15/2000</last_date>
        </delivery>
      </product>
      <id>tr9017-req1</id>
      <last_quote_date>05/1/2000</last_quote_date>
      <payment_type>SAP direct</payment_type>
      <buyer>
          <id>joeinc</id>
          <trading-countries>US</trading-countries>
          <shipping_address>
              <street>1575, Tenaka</street>
              <city>Sunnyvale</city>
              <state>CA</state>
              <zip_code>94087</zip_code>
              <telephone_number>408 733 8463</telephone_number>
          </shipping_address>
      </buyer>
  </ask>
```

- XDisect provides the inherent ability to handle storage and retrieval of varying structures as in the case of the two different product sections above
- XDisect allows for storage and hence matching of RFQs to offers using user-defined keys. E.g. the first product is requested based on the exact part number, whereas for the second product, the customer just knows the type of product and a description and is content with using that for the matching of bids and sellers

Bid

The bid datastructure below represents the seller's response to the above buyer's RFQ. Again based on the privacy requirements of ComponentExchange, the supplier address and related information may or may not be provided to the buyer. Here the bid is for a partial quantity. The seller promises to ship 50,000 parts on 05/20/2000 and the remaining 25,000 parts on 05/29/2000

```
  <bid>
       <id>tr9017-bid1</id>
       <ask_id> tr9017-req1</ask_id>
       <product>
```

```
        <part_num>X232</part_num>
        <line_num>1</line_num>
        <qty>75,000</qty>
        <amount>100000</amount>
        <delivery>
          <ship_date>05/20/2000</ship_date>
          <qty>50,000</qty>
        </delivery>
        <delivery>
          <ship_date>05/29/2000</ship_date>
          <qty>25,000</qty>
        </delivery>
      </product>
     <product>
        <type>4 ohm thermistors<type>
        <line_num>2</line_num>
        <qty>100000</qty>
        <amount>100000</amount>
        <delivery>
          <ship_date>02/15/9000</ship_date>
          <qty>100000</qty>
        </delivery>
      </product>
      <seller>
        <id>chetaninc</id>
        <trading_countries>US</trading_countries>
        <registered_address>
            <street>3386, Valley Forge Way</street>
            <city>San Jose</city>
            <state>CA</state>
            <zip_code>94087</zip_code>
            <telephone_number>408 733 8463</telephone-number>
        </registered_address>
      </seller>
</bid>
```

Detailed Use Cases

In this section we describe some detailed use cases for the above scenario.

Supplier A wants to post/change a product offer

1. Supplier A determines the details of the product offer
2. Supplier A visits the ComponentExchange website using a standard web browser
3. Supplier A does a login into ComponentExchange's website (perhaps by filling out a userid password form or by presenting a digital certificate)
4. ComponentExchange authenticates supplier A against its internally maintained profile

Edit Date	Print Date	**© PyBiz, Inc**	Revision	Page
5/17/00 8:42 PM	05/17/00		0.1	9 of 16
c:\temp\xd_nmm_scenarios.doc				

5. ComponentExchange determines Supplier A's role and preferences from the profile
6. ComponentExchange then displays a personalized page to Supplier A based on the role and preferences
7. Supplier A navigates through the web pages to the product offer online form
8. Supplier A then fills out the offer details and clicks the submit button
9. ComponentExchange collects the form information and automatically converts that into an XML document.
10. ComponentExchange then sends this document to the XDisect engine
11. XDisect engine stores and indexes the product offer document and acknowledges receipt
12. If any events are registered for this change or addition of a product offer, then XDisect will generate the triggers. E.g. notify a customer if a seller is offering CMOS devices
13. ComponentExchange sends back an acknowledgement with a receipt id back to the supplier

- ➢ XDisect provides an open HTTP/XML interface, which makes it convenient for suppliers to directly upload their offer documents into the ComponentExchange Repository.
- ➢ XDisect's HTTP/XML interface allows suppliers to participate in the Market without having to accept and integrate any software from the marketplace
- ➢ XDisect allows the suppliers to post product offers with varying descriptions and yet guarantees that all of them will be considered during an offer selection

Notes :

- Rather than filling out the online form, some suppliers might prefer to directly generate the XML document representing the offer and send it over to ComponentExchange, either via a pre-designated ftp site or via an http post or even email
- If the supplier is not registered, then the login screen could redirect the supplier to a registration request form
- It is assumed that during login session establishment happens between the seller and ComponentExchange

Buyer browses components catalog

1. Buyer visits the ComponentExchange website using a standard web browser
2. Buyer does a login to the website (perhaps by filling out a userid password form or by presenting a digital certificate)
3. ComponentExchange authenticates the buyer by looking up their internally maintained profile and validating their credentials

4. ComponentExchange determines the preferences of the customer from the profile
5. ComponentExchange then queries the XDisect engine for all product offers from all sellers that match the customer's preferences. This could include taking into account approved vendor lists and approved manufacturer lists for the customer
6. XDisect engine searches through all the product offers of the sellers, matches the seller preferences against the buyer's preferences and returns the resulting matching offers. ComponentExchange or the customer can specify the order or sorting sequence for the resulting offers (for presentation)
7. XDisect engine then applies ComponentExchange's business rules for the marketplace to the resulting offers to filter out offers or suppliers that do not match
8. XDisect engine then massages this offer information as per ComponentExchanges sorting and ordering and transformation requirements
9. XDisect engine then paginates the catalog as per the specifications of ComponentExchange (e.g. send over only 25 matches at a time)
10. XDisect engine then invokes the pre-specified ComponentExchange callback module with the results
11. ComponentExchange callback module may converts this XML information into html using XSL or some other means for presentation
12. ComponentExchange then presents this personalized catalog along with the other standard menu pages to the buyer
13. Buyer navigates through the catalog using their web browser
14. Buyer can either drill down through the catalog or may just search it to arrive at the desired component
15. Buyer then selects an offer that seems to match and clicks on a submit button
16.
17. ComponentExchange generates a request for quote or ASK on behalf of the buyer

- ➢ XDisect enables ComponentExchange to create a personalized catalog based on the requirements of the buyer
- ➢ XDisect enables the catalog to be cross vendor. The catalog can contain offer description in varying structures
- ➢ XDisect provides the ComponentExchange the ability to do personalization of buyer user interface by filtering content based on the buyer's profile
- ➢ XDisect can cache the resulting dynamic catalogs for optimum performance
- ➢ XDisect can help transform and aggregate matching offers from various suppliers to convert in to a consistent catalog for presentation to the user

Notes:

- Since the XDisect generated personalized catalog could contain large amounts of information, ComponentExchange software may want to receive the data in specified page sizes
- Since this personalized catalog generation needs to happen frequently, XDisect will cache this catalog based on the request. In addition for better performance, ComponentExchange front end software itself may also want to cache the generated personalized HTML by buyer or seller
- For advanced users, rather than navigating through the catalog , buyer may want to just specify a search query
- Advanced buyers that know exactly what they are looking for could be presented with a ASK submittal form. They can just upload their request for quote or ask XML document and totally bypass the catalog navigation and search portion. The ASK XML document can again be sent over directly to a pre-designated FTP location or via an HTTP post to a designated url

Assumptions :

- ComponentExchange has already defined the business rules for any potential transaction between buyers and sellers
- ComponentExchange has pre-defined the rules for presentation formatting of the data.
- ComponentExchanges could pre-define the rules for sorting and ordering of the result sets. Of course these could be specified with every request or based on the buyer's preferences

Buyer sends an ASK / RFQ

1. Buyer generates an ASK either via navigating through the personalized catalog or manually
2. Buyer then uploads the ASK document either through the website or via ftp to a designated address
3. ComponentExchange receives the ASK XML Document
4. ComponentExchange formulates a query for XDisect engine requesting all sellers that have offers that could match the buyer's requirements taking into account the preferences of the marketplace, the sellers and the buyer

5. XDisect engine searches through its repository and retrives the matching list of sellers
6. XDisect engine then aggregates these sellers and their offers into one consists format
7. ComponentExchange notifies the sellers either manually, via email or via an HTTP Post based on their preferred notification mechanism
8. ComponentExchange sends the ASK XML document along with the associated list of potential sellers to the XDisect engine for storage
9. ComponentExchange registers an event in XDisect requesting that a specified software module be notified whenever a bid is received for the above ASK/RFQ
10. ComponentExchange registers another event in XDisect associated with the ASK /RFQ document requesting that its software module be notified when the specified buyer notification deadline is reached
11. ComponentExchange then sends back an acknowledgement to the buyer stating that it will within X hours or days with the bids

- ➢ XDisect provides an open HTTP/XML interface which makes it convenient for buyers to directly upload their RFQ documents into the ComponentExchange Repository without having to integrate or accept any software from the Exchange
- ➢ XDisect can filter the list of suppliers based on their offers, the preferences of the marketplace, the buyer and the seller. It can then present this filtered list to the marketplace for reverse auctioning
- ➢ XDisect allows lightweight events to be registered against attributes in XML documents. This will enable ComponentExchange to notify interested buyers when sellers are selling certain types of products at a desired price.

Seller responds with a bid

1. Seller gets the RFQ notification via their selected notification mechanism from ComponentExchange
2. Seller then manually or via some automated software agent generates the specified bid, taking into account their current inventory, price requirements, response times, etc. This may involve the seller having to interact with backend legacy systems of their own
3. Seller then uses their preferred interaction mechanism(HTTP, ComponentExchange Web Page, FTP, email) to send the Bid XML document to ComponentExchange.
4. ComponentExchange receives the bid XML document.
5. ComponentExchange forwards this bid document to XDisect engine for storage and processing
6. XDisect engine determines if there are any registered events on the fields in the XML document that should be triggered. In this case it will find the specified ask event that had been registered in the previous use case

7. XDisect engine then notifies the specified ComponentExchange module about the arrival of a new bid
8. ComponentExchange software module will then update a flag in the ASK document indicating that a bid was received from the seller

- XDisect's open HTTP/XML interface makes it convenient for sellers to directly upload their bids into the ComponentExchange Repository without having to integrate or accept any software from the Exchange
- XDisect events make it easy to track all the bids received for any given RFQ
- XDisect events can be used to perform asynchronous processing of bids and RFQs as they are received without having to build in sophisticated workflow or state machine logic in the Exchange

All Bids are received or notification deadline to customer is reached

1. Either the RFQ notification deadline is soon reached or all outstanding bids are received from suppliers. XDisect detects this based on the previously registered events.
2. XDisect engine then notifies the specified ComponentExchange software module about this event and presents a list of the bids received so far
3. ComponentExchange then massages the bids for presentation
4. ComponentExchange then notifies the buyer via their preferred notification mechanism (email, http post, or polling) about the arrival of the bids
5. Buyer then visits the web page (url perhaps embedded in the email) to view the bids
6. Buyer may then place an order against a specified bid or decide to cancel the ask
7. Optionally the buyer may just decide to pick up the phone and call the seller and fax them the bid and the RFQ for order processing

- XDisect lightweight events make it easy to correlate all the bids for a specified RFQ
- XDisect lightweight events make it easy for ComponentExchange to do asynchronous processing of bids notification to the customer without requiring sophisticated workflow or custom event management support
- XDisect's flexible retrieval mechanism make it easy for ComponentExchange to aggregate bids from multiple vendors that could be in diverse and varying formats

Seller receives a big request event notification

1. Buyer sends over an ASK/RFQ for a big dollar amount or quantity
2. ComponentExchange forwards the document to XDisect engine for storage
3. XDisect engine determines that a seller has registered an interest in being notified when an ASK/RFQ is received for the specified big amount
4. XDisect engine then notifies the seller (via their specified notification mechanism) with the details of the big request
5. Seller can then decide to expedite the bid (even before the ask is received) or may start the authentication process for the request or perform any other special action

- ➢ XDisect events make it easy for ComponentExchange to notify sellers when big dollar or exception requests come in. This can allow sellers to do expedited or special processing of these kind of requests without having to write sophisticated software to handle such special cases

Customer receives an interested seller/product event notification

1. Customer is interested in being notified when a seller for certain type of components enters the market
2. Customer registers this interest by filling out a web form on the ComponentExchange website or by posting an XML document for the same directly via ftp or http
3. A seller later on registers a new product offer for the specified type of components
4. XDisect engine receives the product offer, determines that an event has been registered by a customer
5. XDisect then notifies the specified customer about the new seller or the new offer
6. Buyer can then decide to place a request

- ➢ XDisect events make it easy for ComponentExchange buyers to registers events such as " notify me when a certain product at a certain price becomes available in the market"
- ➢ XDisect makes it possible for ComponentExchange to store buyer RFQs that do not match. Eventually when a seller is available, the request can be executed. Thus buyers do not have to constantly keep checking if products of interests at the rates they desire are becoming available

Implications

- XDisect must support synonyms for both paths and words occurring in the data values.

- In some instances XDisect must support Synonyms for a single customer and or a group of customers and or a group of vendors. Think of this as the ability to subscribe to a synonym list.
- The Matching rules for the engine must be fluid enough to detect different conditions and utilize different search techniques. E.G. For part numbers a direct part number against offering look up will be find however for a product type resolution to a set of possible part numbers will be required. The implementation of the exchange must be flexible enough to support external agents in these matching types. Think of this like a very intelligent mime handler where the handler can determine what kind of sub process to spawn for a given data type. In this instance the matching engine will determine which sub type of search agent to utilize in resolution.
-