E-commerce interoperability with IBM's WebSphere Commerce products

by D. M. Dias
S. L. Palmer
J. T. Rayfield
H. H. Shaikh
T. K. Sreeram

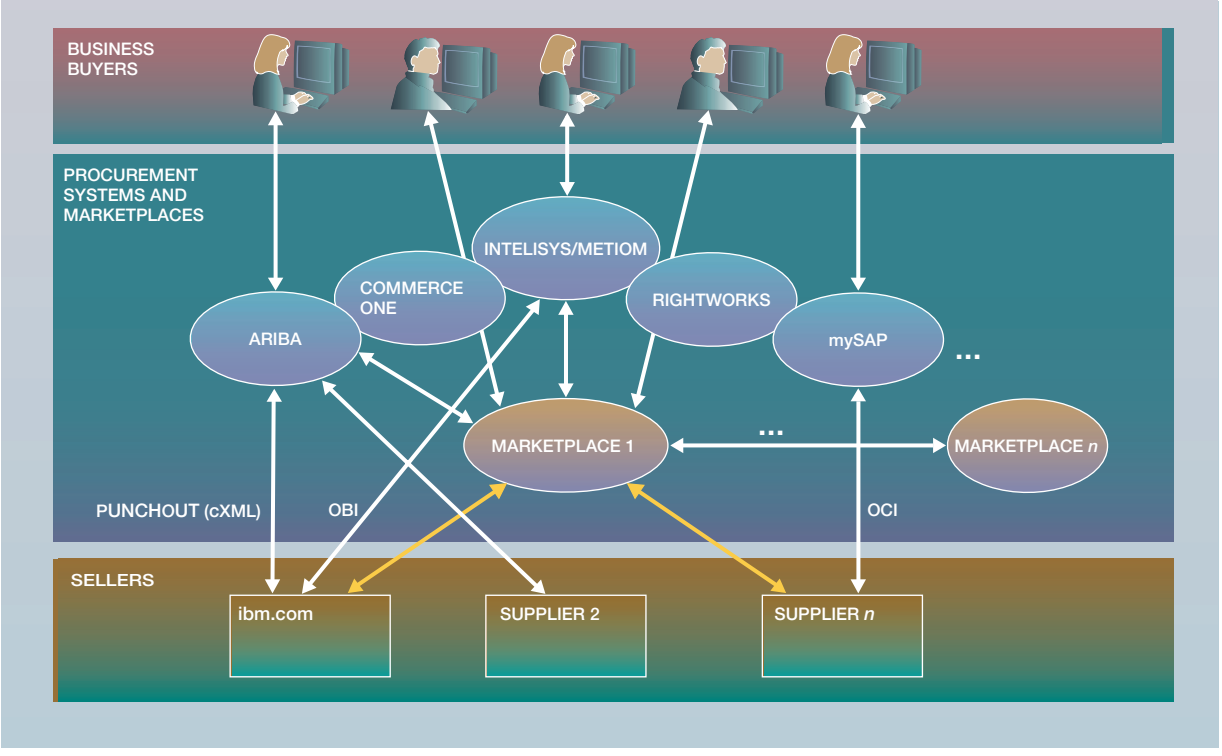
With the growth of the Internet, business-to-business procurement and other processes are being moved to the World Wide Web, for increased efficiency and reach. Procurement systems from different vendors use various protocols, and additional protocols are being defined by several industry consortia. As a consequence, suppliers are faced with the difficult task of supporting a large number of protocols in order to interoperate with various procurement systems and private marketplaces. In this paper, we outline the connectivity requirements for suppliers and private marketplaces, and we describe how suppliers and marketplaces based on IBM's WebSphere® Commerce Business Edition and WebSphere Commerce Suite, Marketplace Edition can interoperate with diverse procurement systems and electronic marketplaces. We first describe simple connectivity based on punchout processes for fixed and contract-based pricing. We then describe how asynchronous processes, such as requests for quote, auctions, and exchanges can be distributed for interoperability across suppliers and marketplaces. Finally, we describe B2B/M2M Protocol Exchange, a prototype we have implemented that maps between different, but analogous, protocols used in procurement systems, and thus alleviates some of the interoperability difficulties.

With the rapid growth of the Internet, organizations are increasingly using the Web to conduct business with greater speed, reach, and efficiency. This transformation is especially prevalent in business-to-business (B2B) commerce and trade. Many of the Fortune 500 companies have adopted e-procurement systems such as Ariba, Commerce One, and mySAP. Many others participate as buyers in e-marketplaces such as Commerce One MarketSet, Ariba Hosted Market Place, and IBM's WebSphere* Commerce Suite, Marketplace Edition (WCS MPE, or MPE for short), among others.

Figure 1 illustrates the environment for B2B procurement on the Web. B2B buyers have diverse procurement systems, such as those offered by Ariba, Commerce One, and SAP, among others. Each of these procurement systems uses different B2B protocols for interaction with seller systems. Many of these protocols are proprietary and specific to the procurement system. For example, as illustrated in Figure 1, Ariba uses the punchout process between the Ariba Order Request Management System (ORMS) and seller systems using their "Commerce XML" (cXML, or Commerce Extensible Markup Language) specification for the messages[1]; Commerce One uses xCBL[2] as the format of messages; mySAP uses the "Open Catalog Interface" (OCI) (for a process similar to punchout) between buyer and seller systems.[3]

0018-8670/02/$5.00 © 2002 IBM

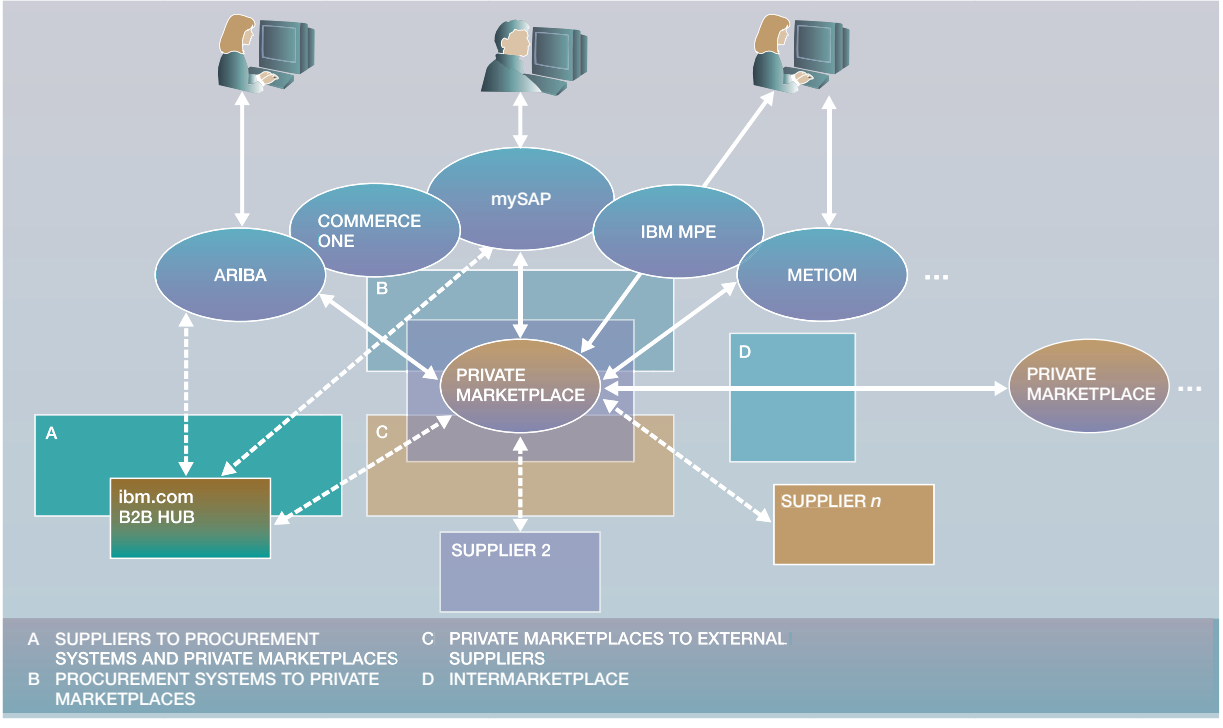
Figure 1 Business-to-business procurement environment



Many other protocols for B2B processes, many proprietary to procurement and other systems, and others customized for specific partners, are being defined and implemented. In addition to the procurement systems, which typically reside within the firewall of the buying organizations, marketplaces are being set up on the Internet through which buyers can access a large number of suppliers, typically for specific industry segments. Many of these marketplaces use the same or similar technology to connect to procurement and supplier systems, and offer buyers at small and medium-sized businesses access to suppliers.

Meanwhile, standards bodies are defining protocols and message formats for B2B processes. One of the early processes was that defined by the Open Buying on the Internet (OBI) consortium,[4] a precursor of the punchout process. The RosettaNet consortium used OBI as a starting point and defined Partner Interchange Processes (PIPs),[5] including both flows and XML-based[6] message formats for interactions between partners. The electronic business XML (ebXML) framework, sponsored by UN/CEFACT (United Nations Center for the Facilitation of Procedures and Practices for Administration, Commerce, and Transport) and OASIS (Organization for the Advancement of Structured Information Standards) includes a messaging service,[7] a Collaborative-Protocol Agreement (CPA)[8] specification, and a Business Processes Specification Schema,[9] all used for enabling the interaction between business processes. The Web services approach[10] defines both a messaging and a remote procedure call mechanism using SOAP (Simple Object Access Protocol);[11] on top of SOAP, the WSDL (Web Services Description Language)[12] defines a Common Object Request Broker Architecture** (CORBA**)[13] IDL (interface definition language)-like interface for Web-based B2B remote procedure calls; and the UDDI (Universal Description, Discovery, and Integration) consortium has defined a directory mechanism for registering and locating businesses on the Web,[14,15] with an optional WSDL interface specification. The Open Application Group (OAG)[16] has defined Business Object Documents (BODs) for the content of B2B messages.

Figure 2 B2B connectivity requirements



Some of these originally disparate efforts are now coming together. For example, the RosettaNet consortium has announced that they will move to the ebXML messaging protocol, and OAG has announced that they will support ebXML. In spite of these efforts, however, the number of B2B protocols continues to grow.

This proliferation of B2B protocols gives rise to several connectivity requirements and problems, as illustrated in Figure 2. First, from a supplier's point of view (box A in Figure 2), suppliers need to connect to the many customer procurement systems and private marketplaces, using various B2B protocols. Second, private marketplaces (and, over time, procurement systems as well) need to connect to procurement systems (box B in Figure 2), using different B2B protocols. Third (box C in Figure 2), private marketplaces need to connect to suppliers where the suppliers may support different B2B protocols. Fourth (box D in Figure 2), private marketplaces need to connect to each other, in order to access suppliers connected to other marketplaces, or to access services offered at other marketplaces.

In this paper, we discuss the connectivity requirements for suppliers and private marketplaces, and describe how suppliers and marketplaces relying on IBM's WebSphere Commerce Business Edition (WCBE) and WebSphere Commerce Suite, Marketplace Edition (WCS MPE) can interoperate within the environment for B2B procurement. The next section describes simple B2B connectivity using punchout processes as supported by WCBE. In the section that follows, we discuss marketplace connectivity for emerging asynchronous processes and distributed trading mechanisms, as supported by WCS MPE. In the section "Connectivity using a B2B protocol exchange," we show that many of these protocols can be mapped to each other, thus allowing procurement systems and suppliers to use possibly different protocols. The last section contains ideas for future work and concluding remarks.

Simple B2B connectivity using punchout

In this section we focus on two of the B2B connectivity problems mentioned above and illustrated in Figure 2. First, we discuss the supplier connectivity

problem and present a solution based on IBM's WebSphere Commerce Business Edition[17] (WCBE) for connectivity of suppliers to diverse procurement systems. Second, we discuss marketplace connectivity and present a solution based on IBM's WebSphere Commerce Suite, Marketplace Edition[18] (WCS MPE) for connectivity of marketplaces to diverse procurement systems and diverse supplier systems.

Most procurement systems and private marketplaces support the notion of punchout (albeit using sometimes a different term, such as RoundTrip, used by Commerce One). A buyer at a procurement system or marketplace selects a remote supplier, the buyer is automatically logged on to the supplier catalog server and presented with a catalog customized for his or her organization, with prenegotiated prices. The buyer shops at the site, the items selected for purchase being stored in a shopping cart. On checkout the shopping cart contents are sent back to the buyer's procurement system for approval. The procurement system provides workflow for approvals and, on approval, a purchase order is sent from the procurement system to the supplier. Additional messages may be exchanged between the supplier and the procurement system, such as shipping notices and invoices. (Details of the punchout flow are provided later.) By having punchout capability, suppliers and marketplaces can interoperate with procurement systems or marketplaces, with significant benefits to both suppliers and buyers.

IBM WCBE version 5.4 is a solution for the business-to-consumer trade, whereas WCS MPE version 4.2 supports the private trading exchange customers. Customers can connect to the WCBE Web site, browse through the catalog, and place orders. In the case of WCS MPE, customers have the benefit of working with various trading mechanisms such as RFQs, auctions, reverse auctions, and exchanges. It is especially useful, given the emerging trends in the industry, that the WebSphere Commerce products have punchout capability and be able to interoperate with buyers' procurement systems and marketplaces.

Although WCS MPE supports aggregation of suppliers' catalogs, certain suppliers may have enormous catalogs and their systems may include complex configuration tools; often it is not feasible to offload supplier catalogs into external marketplaces. Thus suppliers often have their supply-side Web sites enabled for punchout, and expect WCS MPE to initiate punchout to the supplier Web site.

Catalog aggregation in the current WCS MPE product is done using the WebSphere Catalog Manager (WCM) product. WCM supports loading of catalog data into an electronic marketplace (eMP) database, transforming catalog data from ASCII, spreadsheet, and XML formats into a canonical XML format, and extracting catalog data from any relational database. More enhancements to support industrial catalogs are planned for future versions of WCM.

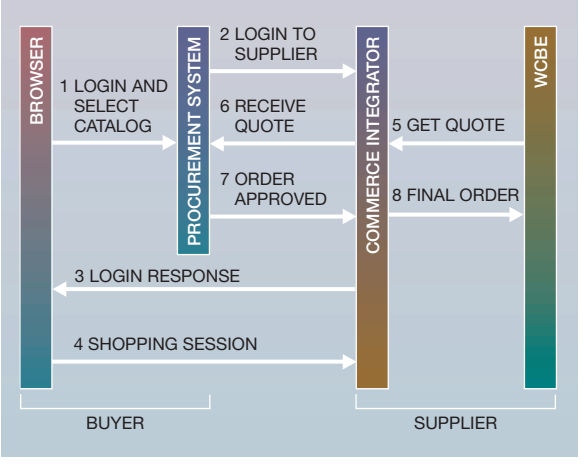
Many large corporations have relatively independent subsidiaries and are classic examples of customers that require support for both receiving punchout requests and initiating punchout requests. Such corporations often have aggregated supplier catalogs across their subsidiaries so that their customers see a unified company-wide catalog and require support for receiving punchout requests from the buyers' procurement systems to the aggregated catalog. They also require punchout initiation functionality to connect from their aggregated-catalog server to individual catalogs supported by their subsidiaries.

Punchout from procurement systems to WCBE and WCS MPE. IBM Commerce Integrator is a generic framework that enables WCBE 5.1 and WCS MPE 4.2 to handle business-to-business transactions using industry standard protocols. It offers customers the opportunity to integrate their systems with the procurement system's own network of high-volume buyers. Commerce Integrator provides an integrated, scalable system that enables suppliers with WCBE to participate as a supplier in the procurement system's marketplace, to increase sales, and to enhance their business-to-business presence on the Web. Specifically:

- Suppliers maintain a single catalog within WCBE and use that catalog to enable their own Web presence as well as to participate in the procurement system's network.
- Suppliers can take advantage of WCBE connectivity to supply chain management systems, retail business systems, and order management backend systems to automatically flow orders from the buyer's procurement system.
- Suppliers can take advantage of the updated business-to-business features of the WCBE product for using and maintaining information about buyer organizations, buyer-specific catalogs and price lists, and contract pricing.

Figure 3 illustrates a high-level view of a typical punchout flow in which WCBE interoperates with an

Figure 3 Typical punchout flow using WCBE and Commerce Integrator



e-procurement system, which includes the following steps.

1. An agent in the buyer organization logs on to the procurement system using the user ID (identifier) and password, and then selects an external catalog. The procurement system authenticates the buyer agent.
2. The procurement system constructs a request to access the external supplier catalog using a user ID and other buyer organization credentials.
3. The Member Subsystem of Commerce Integrator authenticates the buyer agent against the buyer organization data stored in the WCBE database. If successful, the buyer agent is presented with a catalog customized for the buyer organization.
4. The buyer agent browses the catalog in the WCBE database while a shopping cart is created. On checkout, the shopping cart is submitted to WCBE, and a quote is recorded in the database.
5. Commerce Integrator picks up the quote from WCBE.
6. Commerce Integrator sends the quote to the buyer in the format required by the buyer's procurement system. An authorized agent for the buyer is prompted for acceptance of the quote.
7. The authorized agent approves the quote. An order from the procurement system is sent to Commerce Integrator.
8. Commerce Integrator forwards the order to WCBE.

Further messages, such as advance shipment notices and invoices (not shown in Figure 3) are sent from WCBE to the procurement system.

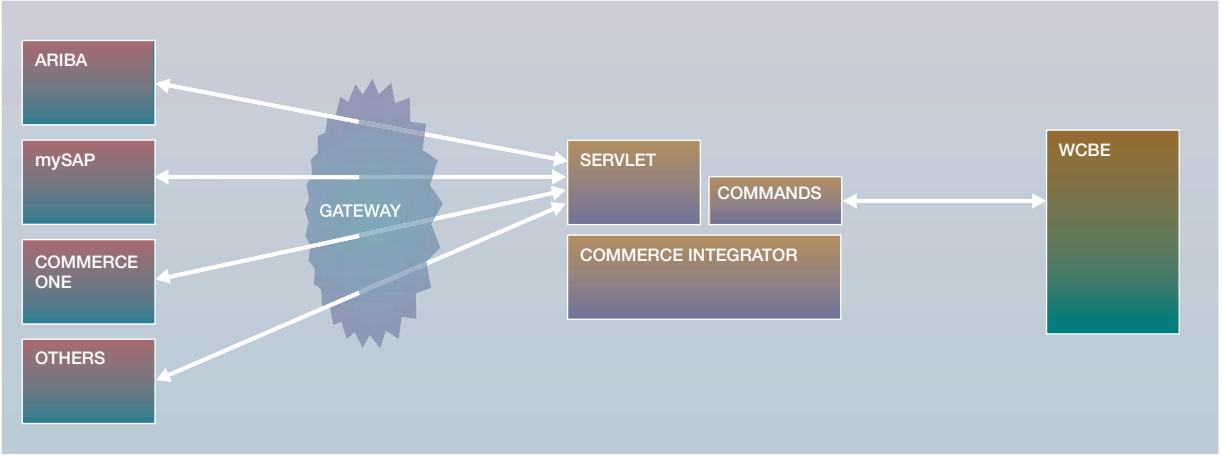
Although the punchout flow is similar for most procurement systems, the message format is different for different procurement systems. For example, Ariba uses cXML messages,[1] mySAP uses HTML name-value pairs,[3] Metiom uses the OBI EDI (electronic data interface) message formats,[4] and Commerce One uses xCBL message formats.[2] There are some differences between the flows, as outlined in the section on the B2B protocol exchange. To handle these differences, Commerce Integrator includes some protocol-specific functions, in addition to functions common to all protocols. As shown in Figure 4, incoming messages are handled by a common servlet, which identifies the protocol and calls protocol-specific functions that map the message to a common internal format. Then WCBE commands, shared by all punchout protocols, are invoked. Responses are converted from the common format into protocol-specific formats by Commerce Integrator.

Figure 4 also shows a B2B gateway. The function of the B2B gateway is to provide a means of connecting remote trading partners over the Internet, each using its protocol of choice. Clearly, this functionality facilitates the integration of interenterprise business processes. Although the B2B gateway may support additional functions, such as business process management, audit trails, and intraenterprise connectivity, we do not elaborate further on these functions.

The protocol associated with an incoming message is identified by the URL (uniform resource locator) to which the request is sent. The use of a single servlet for all requests should have no negative performance impact, because the servlet engine launches a new thread for each request. Performance bottlenecks would only be caused by undue contention for shared resources. Were such contention present, it would impact multiple servlets in the same manner as a single servlet. Because the servlet is merely the entry point for requests that quickly fan out to different parts of the server, it is unlikely that the degradation of reliability from the use of a single servlet would be significant.

There are two scenarios of interest: one in which there is no separate B2B gateway, and one in which there is a gateway present. When there is no B2B gateway, protocol-specific requests are sent to Commerce Integrator, and appropriate commands are invoked.

Figure 4 WCBE Commerce Integrator architecture



If a B2B gateway is present, the incoming requests are mapped into a common canonical format, and then Commerce Integrator invokes appropriate WCBE commands. Thus, there is a synergistic relation between WCBE/Commerce Integrator and the gateway.

Punchout from WCBE and WCS MPE to external suppliers. A traditional electronic marketplace (eMP) or a private trading exchange (PTX), such as IBM WCS MPE 4.2, provides various trading mechanisms: RFQs, contract-based buying, fixed-price buying, auctions, exchange, etc. It also provides support for aggregated catalogs. Both buyers and sellers begin by using the catalog to select a product to buy or to sell. When sellers offer products for sale, they specify the method of purchase to be used: RFQ, contracted price, fixed price, auction, or exchange. Buyers must purchase products using the method specified by the seller (with the exception of RFQ, which they can initiate).

Aggregating the catalog at the eMP site offers advantages including:

- It makes possible a parametric search across suppliers.
- It enables small businesses, which do not have the infrastructure to host catalogs, to engage in e-commerce.

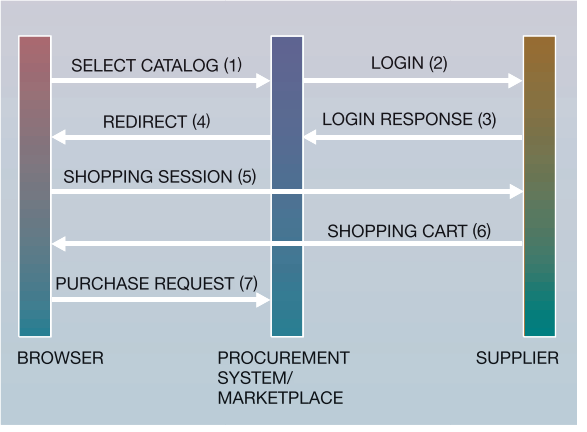
Aggregating catalogs, however, has its own limitations, including:

- It does not preserve each supplier's unique brand and Web site design (it requires direct links to the supplier's Web pages).
- It supports only static content rather than promoting dynamic, up-to-date information.
- It provides limited support for suppliers with very large catalogs.
- It provides no support for product configurators (needed for complex products).
- It provides limited support for suppliers with fast changing catalogs or pricing.

Thus, in situations where there is a need for product configurators, or if the catalog contains fast changing products and prices, the suppliers have to maintain catalogs at their own sites and not aggregate the catalog onto an eMP. In the common eMP approach, a buyer has access to only the sellers who participate in the marketplace with which the buyer is registered. Similarly, a seller cannot sell goods and services in a marketplace different from the one with which the seller is registered. We describe next a mechanism called punchout in which a buyer in a private marketplace can "punch out" to a remote supplier to buy fixed-price and contract price offerings.

Figure 5 shows the flow for setting up a punchout process from a procurement system (or marketplace) to a supplier site, for example, a WCBE site. Remote suppliers are listed at the procurement system. They may provide their entire catalog remotely using punchout. Alternatively, a supplier may provide a

Figure 5 Typical punchout request flow



local catalog at the procurement site, with links for specific functions or details. For example, a supplier may use punchout for system configuration, or for parts of the supplier catalog that may change frequently. As shown in Figure 5, after selecting a remote supplier for initial or further shopping (1), a login request (2) is sent to the remote supplier as an XML document, encapsulating the user and organization credentials, as well as a URL for postback to the procurement system (used at step 7, below). The remote supplier authenticates this request and returns a URL (3) with embedded user information. The client's browser is redirected (4) to this URL, allowing the buyer to directly shop (5) at that remote site using the appropriate catalog for the buyer's organization. At the end of the shopping session, a quote representing the shopping cart is sent back to the client (6) and posted back to the procurement system (7) at the postback URL referred to above.

After the purchase request (in XML format) is received back at the procurement system (7), it is parsed and added to the buyer's requisition. (The buyer may punchout to multiple suppliers and add the contents of those shopping carts to his or her requisition.) The buyer then submits the requisition for approval. After submission, the buyer can then view the submitted requisition and its status, and modify the requisition, if so desired.

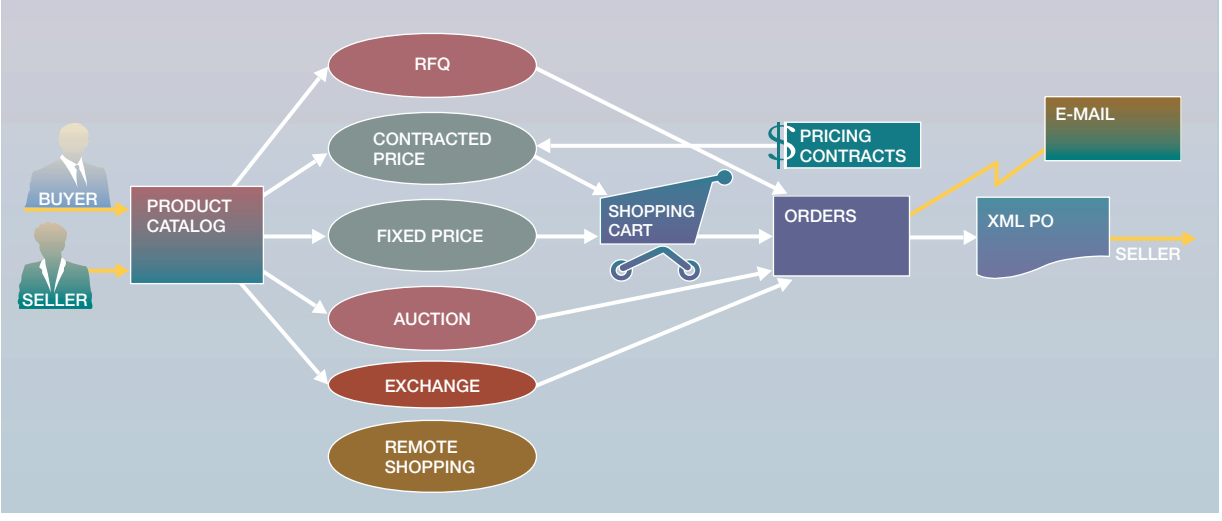
Subsequently, the approver views the submitted requisitions, and optionally may punchout to the supplier to view details of the requisition. The approver can modify the requisition, if so desired. If the approver rejects the requisition, the status is so indicated, and can be viewed by the buyer; if the requisition is approved, it is converted into one or more purchase orders (POs), and sent to the supplier(s). The PO is sent as an XML document, in the format required by the supplier. If the remote supplier's system is based on WCBE, the PO is formatted in a common canonical format; if it is an Ariba-compliant supplier, it is formatted in cXML. If the format is different, a B2B protocol exchange can be used to convert the PO to the desired format and protocol. When the remote supplier acknowledges the receipt of the PO, the state of the order at the procurement system is updated. Subsequently, additional messages may be sent by the supplier to the procurement system to indicate further events, such as issuing an advance shipping notice.

Marketplace connectivity for asynchronous processes

As illustrated in Figure 6, IBM's WCS MPE 4.2 provides different trading mechanisms such as fixed-price buying, contract-based buying, RFQs, auctions, and exchanges. In the previous section, we described the punchout mechanism that can be used for remote supplier integration when dealing with fixed and contract pricing. However, the more advanced trading mechanisms, including RFQs, auctions, and exchanges, cannot be supported by the basic punchout mechanism. This is because the flows between WCS MPE and the remote suppliers for fixed and contract pricing are synchronous, and occur during a real-time session with the buyer, making them amenable to the on-line punchout process. RFQs, auctions, and exchanges involve asynchronous interactions between WCS MPE and the supplier. In this section we describe how such asynchronous processes are handled. We use RFQs as a typical example; similar flows and XML document interchanges can be used for other asynchronous trading mechanisms.

In WCS MPE, an RFQ is a trading mechanism used when a buying organization attempts to obtain a special price for a purchase, or when a buying organization cannot find an acceptable offering in the eMP aggregated catalog that meets its needs. The RFQ may be issued in order to obtain a special price based on quantity for well-defined items or for a group of items. The RFQ may also be issued for unique items based on the buyer's description. The request is sent to one or more selling organizations, and these may submit a bid on the RFQ. The selling organizations

Figure 6 Trading mechanisms in WCS MPE



respond to the RFQ and the buying organization may select one or more winning responses. The result of the RFQ process could be an order placed by the buyer or a contract could be created for the negotiated price. Figure 7 shows this process flow in WCS MPE.

We now describe two different mechanisms for extending the RFQ process to a distributed environment. The first mechanism, which we refer to as "local RFQ," exploits the advantages of aggregating the catalogs at the eMP site, while distributing only the RFQ process. The second mechanism, which we refer to as "remote RFQ," allows buyers to connect to a remote WCBE at a supplier or a remote WCS MPE and issue an RFQ.

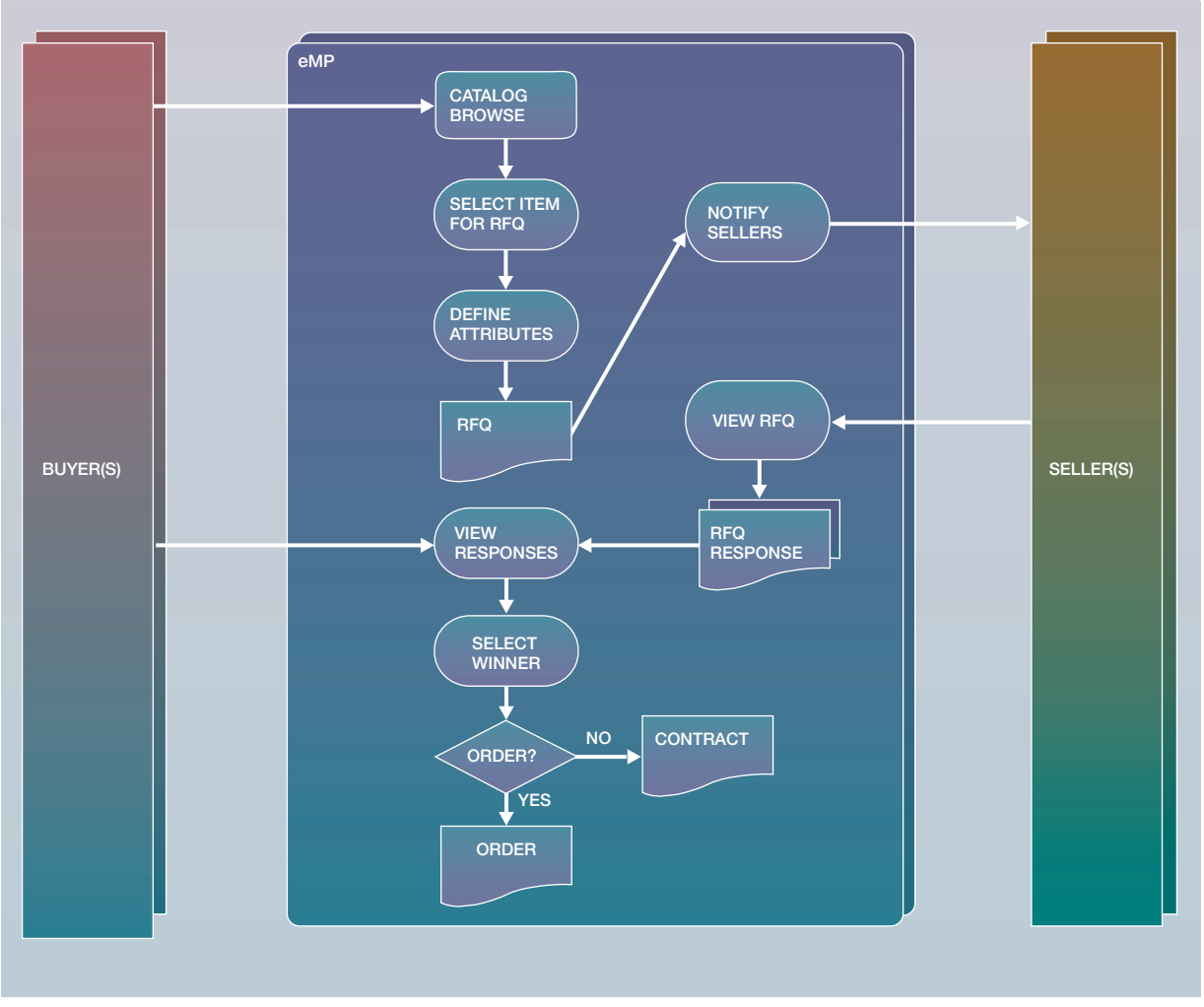
For local RFQs, the catalog is hosted at the WCS MPE site where the buyer is registered. Figure 8 shows the process flow for this configuration. The configuration includes the following parties:

- One or more buyers
- An eMP where the buyers are registered
- One or more remote eMPs
- One or more sellers registered on the remote eMP

The flow starts with the buyer browsing the catalog on the eMP and creating an RFQ. The RFQ is sent as an XML message to the remote eMP. Upon receiving the RFQ, the remote eMP notifies the target sellers. Each seller views the RFQ and creates a response for it. The asynchronous responses are then sent to the eMP as XML messages. The buyer can check the status of the RFQ at any time. The buyer views the RFQ responses by logging on to the eMP, evaluates them, and selects a winner. Selecting a winner leads either to a purchase order or a negotiated contract. The order or the contract is then sent to the remote eMP or remote seller as an XML message. This solution has the advantages of an aggregated catalog and allows buyers on one eMP access to sellers on a remote eMP, and vice versa. It has, however, the previously mentioned limitations of aggregated catalogs.

For remote RFQs, the catalog is hosted either on the remote eMP where the seller is registered, or on the remote seller's Web site. Figure 9 shows the process flow for this configuration. This configuration also involves four parties. The flow starts with the buyer selecting on the local eMP a registered remote eMP or a remote seller. The eMP connects the buyer to the remote eMP site. The buyer browses the catalog on the remote eMP and creates an RFQ template. The RFQ template is then sent as an XML message to the eMP. The RFQ template received from the remote eMP is converted into RFQ by providing additional information. It can then be optionally submitted for approval. Finally it is sent to the remote eMP or remote seller as an XML message. The remote eMP notifies the target sellers. The sellers view the RFQ and create responses for it. The responses are then sent

Figure 7 RFQ process flow in WCS MPE



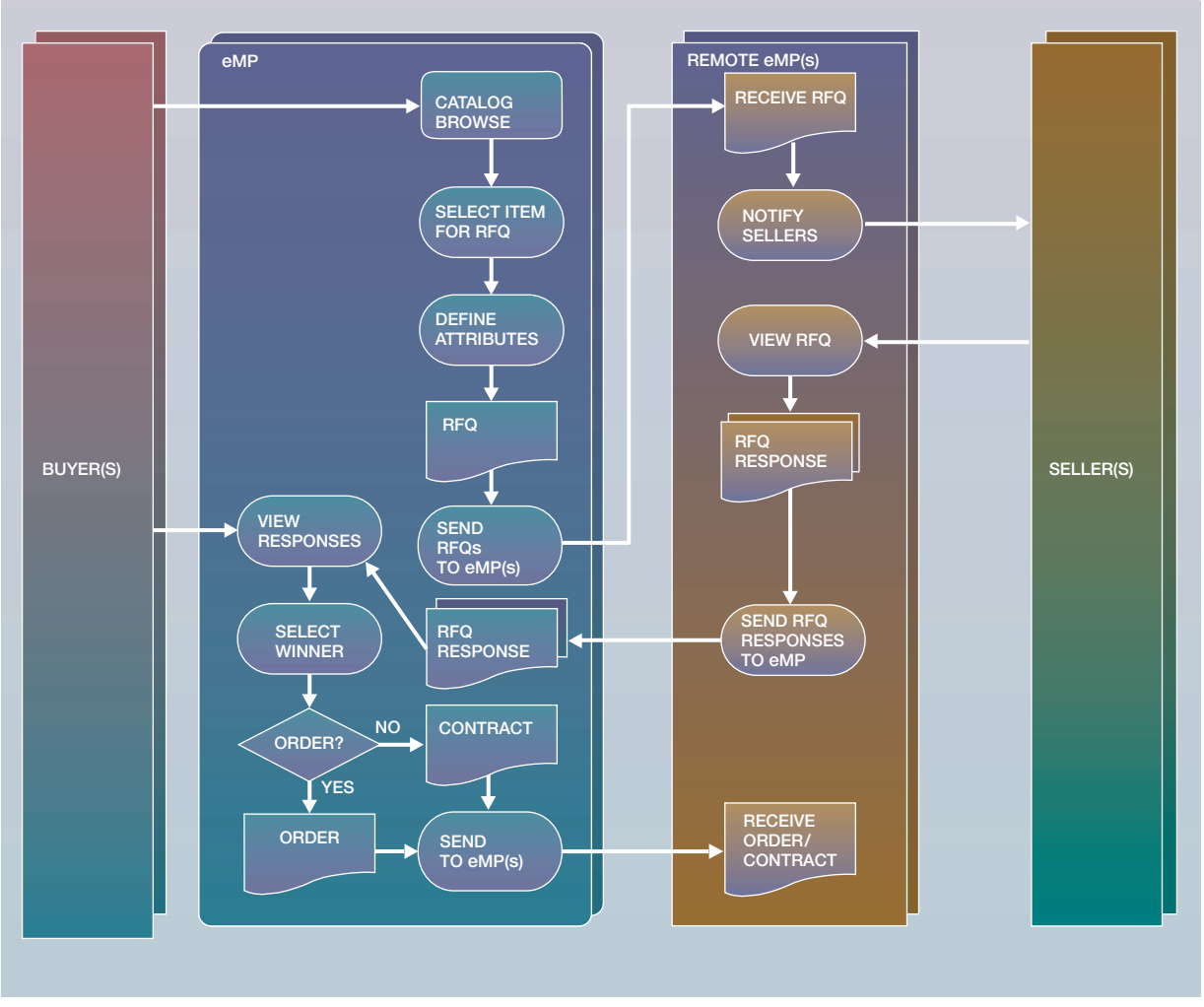
to the local eMP as XML messages. The buyer views the RFQ responses by logging on the eMP, evaluates them, and selects a winner. Selecting a winner leads either to an order or to a negotiated contract. The order or the contract is then sent to the remote eMP or remote seller as an XML message.

This solution overcomes the limitations of aggregated catalogs for such asynchronous trading mechanisms, and allows buyers on one eMP access to sellers on a remote eMP, and vice versa. This comes at the price of losing the advantages of aggregated catalogs.

Connectivity using a B2B protocol exchange

In a previous section we touched on the fact that some suppliers participating in a private marketplace prefer to keep the catalog contents to themselves and not participate in an aggregated catalog hosted by the marketplace. As B2B connectivity becomes increasingly popular, the number of protocols for engaging in B2B transactions continues to grow. Given this growing "babelization" it is likely that businesses and marketplaces that need to communicate will be using different protocols. For this reason we have built B2B/M2M Protocol Exchange (M2M stands for

Figure 8 RFQ process flow for local RFQ



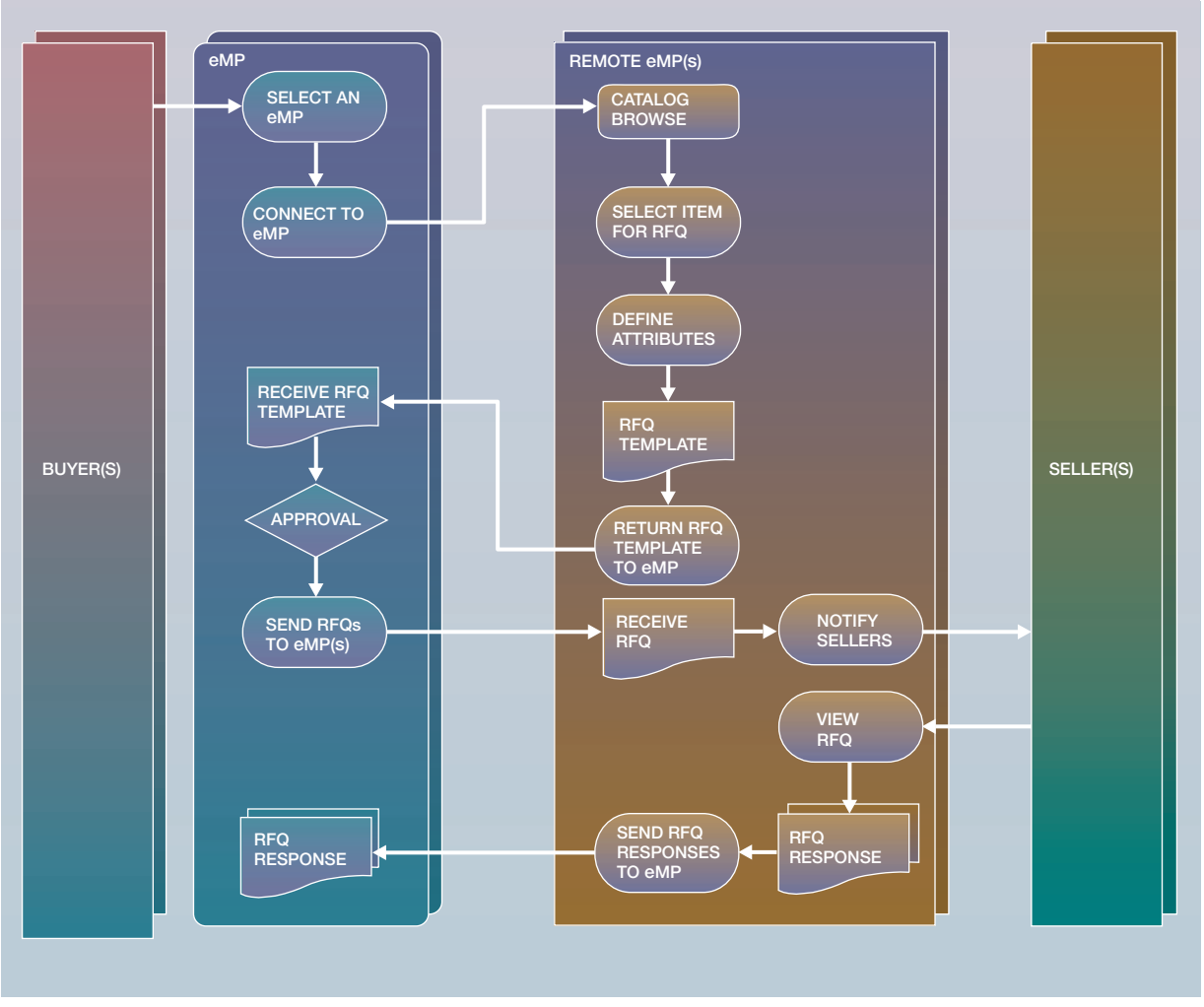
market-to-market), a prototype capable of converting between different protocols.

In this section we first describe how the exchange could be used to enable punchout between a buyer and a supplier using different protocols. Although this example is limited to punchout, the protocol exchange can cover many other common B2B interactions such as shopping cart processing and order processing.

Suppliers participating in a marketplace may have catalog systems already in place supporting existing standard or proprietary formats. These formats may vary from supplier to supplier. Thus, Supplier A may support cXML[1] punchout messages, Supplier B may support OCI[3] punchout messages, and Supplier C may support some other format. The marketplace punchout function must send punchout messages in the format and protocol that a specific supplier is capable of processing. The B2B protocol exchange is a tool that allows suppliers to interact with buyers whose protocols would otherwise be incompatible.

Unlike some kinds of protocol conversions, most B2B protocol conversions cannot be achieved in a *stateless* manner, that is, in a manner in which the protocol converter has no knowledge of prior events or

Figure 9 RFQ process flow for remote RFQ



message exchanges. This is because many of the protocols refer to the session state or to prior messages. In other words, a B2B protocol involves not only message formats but also message flow and the state of the interchange process between business partners. Thus, session state management is required along with message format translation.

A block diagram of a typical environment is shown in Figure 10. In this illustration, Buyer 1 and Supplier 1 use protocol A, whereas Buyer 2 and Supplier 2 use protocol B. Information exchange between Buyer 1 and Supplier 2, or between Buyer 2 and Supplier 1, requires the use of the protocol exchange. The presence of the exchange is transparent to buyers as well as suppliers. When Buyer 1 and Supplier 2 are interoperating, Supplier 2 appears to Buyer 1 to be a protocol A supplier, and Buyer 1 appears to Supplier 2 to be a protocol B buyer.

We now describe in some detail a punchout operation such as an Ariba cXML punchout between a buyer and a supplier that use the same protocol. The data flow is illustrated in Figure 5, shown earlier; the numerals refer to the process steps described here. To purchase from a network catalog, the buyer typically uses a browser to interact (1) with the procurement system, and through the procurement system

establishes a connection to a network catalog hosted on the supplier's behalf. The procurement system thus sends a login request (2) (e.g., a cXML PunchOutSetupRequest) to the supplier system. The login request contains the credentials (e.g., userid/password) of the procurement system, a session identifier (e.g., ⟨BuyerCookie⟩ in cXML), and the postback URL, which is the HTTP URL at which the procurement system accepts the completed purchase requests (in step 7, below). The supplier system authenticates the request and responds (3) with the URL for accessing the network catalog (e.g., in a cXML PunchOutSetupResponse). The procurement system then redirects the browser to the network catalog URL (4), and the buyer connects directly to the network catalog system (5), bypassing the procurement system.

We have previously described in some detail the punchout operation, illustrated in Figure 5, between a buyer and a supplier that use the same protocol. In the case where the buyer and supplier use different protocols, they will be unable to support a punchout interoperation unless some mechanism such as the protocol exchange is used. The data flow is shown in Figure 11. When using a protocol exchange for this mapping, the procurement system is configured to treat the exchange as the supplier system. The initial login request (2a) is sent to the exchange rather than the target supplier system. The processing required at the exchange at this point may be fairly involved. Typically the protocol conversion involves two different authentication domains (the source protocol and the target protocol). The exchange must validate the incoming credentials and generate the outgoing credentials for the target protocol domain. In addition, the incoming request typically has an associated session ID (e.g., BuyerCookie), which must be recorded and mapped to an equivalent session ID in the target protocol. Also, the postback URL must be saved, and the URL of the exchange must be substituted in the outgoing message. Finally, the target supplier system must be identified, and the converted request must be passed as a new login request (2b) to the target supplier system.

When the login response (3a) is received by the exchange, the response is converted into a protocol A response by the exchange and returned to the procurement system (3b). The procurement system redirects (4) the browser to the network catalog site, and the shopping session (5) takes place directly between the buyer's browser and the network catalog site. At checkout time, the browser accepts the contents of the shopping cart in protocol B format (6),

Figure 10 Typical B2B environment using protocol conversion

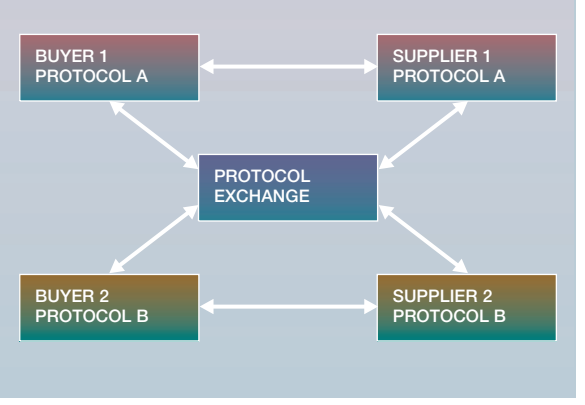
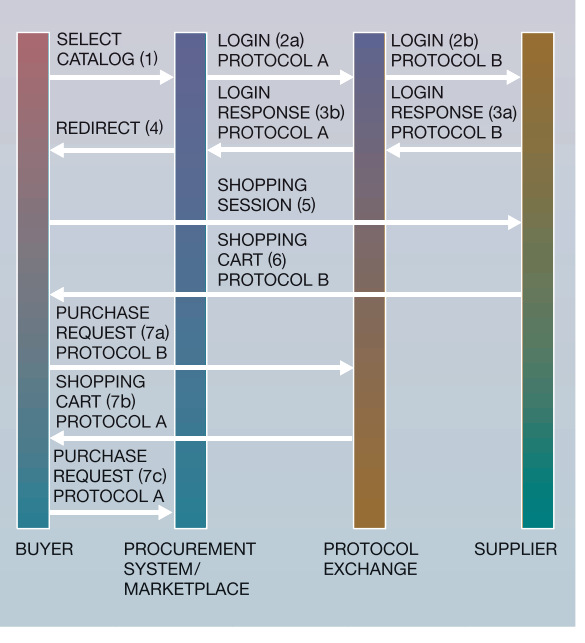


Figure 11 Punchout request flow with protocol conversion



and sends it to the exchange (7a) rather than to the procurement system, due to the substitution of the exchange URL for the procurement system URL in the protocol A login response. In order to process the checkout, the exchange creates a new checkout page, with the shopping cart converted into the protocol A format, and returns this page to the buyer's browser (7b). The target URL of the "checkout" but-

ton on this page is the postback URL of the procurement system, which was saved during the translation of the login request in step (2a). The buyer is instructed to perform a second checkout operation (7c), which causes the purchase request to be submitted to the procurement system for approval. The second checkout may be hidden from the user by using scripting (e.g., JavaScript**) in the HTML page generated by the exchange.

This particular punchout description is one example of how the exchange flows might operate. Specific protocol flows will vary in the exact details. The protocol exchange run time is constructed from a set of common protocol objects (e.g., Login, ShoppingCart, Order), with plugins for specific functions of the various protocols. For example, the mySAP inbound logon plugin accepts a mySAP logon request and converts it to an internal logon object. The cXML outbound logon plugin converts the logon object into a cXML PunchOutSetup Request. The various shopping cart plugins convert shopping carts in different protocols into a common ShoppingCart object. The exchange also contains code to map between credential domains, e.g., from Ariba Network IDs to mySAP OCI userid/password. Finally there is a state management framework to maintain the state of a session and keep track of message content (such as the postback URL), which must be extracted from one message, temporarily saved, and replaced in a subsequent message.

The B2B interaction between two parties is defined within the protocol exchange as a series of plugin transformations to be performed. One plugin will accept a message and turn it into a common object. A subsequent plugin will take the object and issue it as a message in a different protocol. There is no implicit assumption, for example, that a cXML punchout to a supplier will result in the supplier returning the shopping cart in cXML format, or that a shopping cart returned in cXML format is to be followed by an order to the supplier in cXML.

This flexibility is necessary to accommodate some of the interactions that are common today. As an example, the SAP Open Catalog Interface[3] allows the shopping cart to be returned in either XML or HTML, depending on the configuration of the buyer's procurement system. Some of the private buyer and supplier marketplaces are implemented using combinations of different protocols. A supplier might expect an OBI logon from which it might return a cXML shopping cart to the purchasing system. And the subsequent order may have to be transmitted in EDI because the supplier's EDI order processing system was in place running over a value-added network long before the supplier had implemented any B2B interactions over the Internet.

It is hoped the various electronic commerce dialects will someday coalesce into a smaller and more concise set. But until then, it seems that something like a B2B protocol exchange will be required to bridge the communication gap between prospective trading partners.

Summary and conclusions

In this paper, we have outlined various business-to-business connectivity protocols between procurement systems, private marketplaces, and suppliers, and have described how WCBE-based suppliers and private marketplaces can connect to diverse procurement systems, other suppliers, and external private marketplaces. Specifically, we showed how WCBE-based suppliers and WCS MPE-based marketplaces can connect to buyers at procurement systems that use punchout, such as Ariba,[1] Commerce One,[2] and mySAP.[3] We then described how a WCS MPE-based supplier or private marketplace could originate a punchout process in order to connect to either an external supplier or another private marketplace.

Next, we outlined the types of trading mechanisms that can be supported by existing punchout protocols and the asynchronous trading mechanisms, such as RFQs, that require extensions to the punchout mechanisms. While these mechanisms can be used across WCS MPE-based suppliers and private marketplaces, such mechanisms need to be standardized in order to enable them to connect to suppliers and marketplaces provided by other vendors.

Finally, we described B2B/M2M Protocol Exchange, a tool we have implemented that can map between various protocols used by different procurement systems. It allows a supplier using one protocol to connect to a procurement system or private marketplace that uses a different protocol.

The WCBE-based Commerce Integrator, with support for B2B procurement protocols as described in this paper, has been used to connect ibm.com, as a supplier, to enterprises using diverse procurement systems and to private marketplaces. Although, in this paper, we have focused on the external partner B2B protocols, a large part of the integration effort for suppliers is the tie-in to internal processes, such

as the processes to handle purchase orders. Other complementary products, such as IBM's MQSeries (to be renamed WebSphere MQ) and WebSphere Business Integrator, are key to completing the picture for end-to-end integration.

Acknowledgment

We would like to thank Thao Nguyen, Anthony Tjong, Sreedhar Janaswamy, Jojo Joseph, N. Krishnan, and Catherine Crawford for their contributions in building some of the functions described in this paper and their use in customer situations.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Sun Microsystems, Inc.

Cited references

1. *Commerce Extensible Markup Language (cXML) Users Guide*, Version 1.2, Ariba Inc., http://www.cxml.org (2001).
2. *XML Common Business Library (xCBL) Element and Datatype Structure Reference*, Version 3.0, Commerce One, Inc., http://www.xcbl.org (2001).
3. Open Catalog Interface (OCI), Release 2.0 B, SAP AG, http://www.sap.com/partners/software/integration-opportunities/interface-certification/srm/b2b-oci.asp (2000).
4. *Open Buying on the Internet Technical Specifications*, Release 2.0, The Open Buying on the Internet (OBI) Consortium, http://www.openbuy.org (1999).
5. *RosettaNet Partner Interface Process (PIP) Technical Architecture*, RosettaNet Consortium, http://www.rosettanet.org (1998).
6. Extensible Markup Language (XML), 1.0, World Wide Web Consortium, http://www.w3c.org/XML (1998).
7. *ebXML Message Service Specification*, Version 1.0, UN/CEFACT and OASIS, http://www.ebxml.org/specs/ebMS.pdf (2001).
8. *Collaboration-Protocol Profile and Agreement Specification*, Version 1.0, UN/CEFACT and OASIS, http://www.ebxml.org/specs/ebCCP.pdf (2001).
9. *ebXML Business Process Specification Schema*, Version 1.01, UN/CEFACT and OASIS, http://www.ebxml.org/specs/ebBPSS.pdf (2001).
10. Web Services Zone, IBM developerWorks, http://www-106.ibm.com/developerWorks/webservices/.
11. Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP/.
12. *Web Services Description Language (WSDL) 1.1 Specification*, World Wide Web Consortium, http://www.w3.org/TR/wsdl (2001).
13. Common Object Request Broker Architecture (CORBA), Object Management Group (OMG), http://www.omg.org/.
14. *Universal Description, Discovery, and Integration (UDDI) Programmer's API Specification*, Version 2.0, UDDI.org, http://www.uddi.org/pubs/ProgrammersAPI - V2.00 - Open - 20010608.pdf (2001).
15. *Universal Description, Discovery, and Integration (UDDI) Technical White Paper*, UDDI.org, http://www.uddi.org (2000).
16. Open Applications Group (OAG), http://www.openapplications.org/.
17. See http://www-3.ibm.com/software/webservers/commerce/wc_be/.
18. See http://www-3.ibm.com/software/webservers/commerce/wcs_be/index.html.

Accepted for publication November 6, 2001.

Daniel M. Dias *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (electronic mail: dias@us.ibm.com).* Dr. Dias received his Ph.D. degree in 1981 from Rice University, Houston, TX. At IBM Research he manages the Parallel Commercial Systems department, where ongoing projects focus on business-to-business e-commerce, distributed middleware, Web services, and high-volume Web serving. Technologies developed in these projects have been used at the Sydney Olympics Web site as well as large customer sites, and some are now available as IBM products, such as Network Dispatcher, iSeries™ Connect, and Web Cache Accelerator. Dr. Dias's recent work involves scalable and highly available Web servers, frameworks for business-to-business e-commerce, high-performance scalable Web caches, highly available clustered systems, and performance analysis. He has published more than 100 papers in refereed journals and conferences. Dr. Dias has won two best paper awards, IBM Outstanding Innovation and Technical Achievement Awards, and 12 invention achievement awards.

Stewart L. Palmer *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (electronic mail: slp@us.ibm.com).* Mr. Palmer is a senior software engineer at the IBM Thomas J. Watson Research Center. His current research interest is in the area of reliable business-to-business e-commerce. He is the principal architect of the B2B protocol exchange described in this paper. He has an extensive background in systems software development projects including operating systems for Unisys and IBM, DB2® (Database 2™) for OS/390®, and messaging middleware, and he has made contributions to MVS™, and to System/390® architecture.

James T. Rayfield *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (electronic mail: jtray@us.ibm.com).* Dr. Rayfield is a research staff member at IBM Research. He received his Ph.D. degree in 1988 from Brown University, Providence, RI. His research interests include object-oriented transaction-processing systems and database systems.

Hidayatullah H. Shaikh *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (electronic mail: hshaikh@us.ibm.com).* Mr. Shaikh is a senior software engineer in the Parallel Commercial Systems department at the IBM Thomas J. Watson Research Center. His current research area is business-to-business e-commerce, focusing on defining a flexible and scalable framework for new and existing business-to-business protocols. His contributions include the ebXML header specification and tpaML, the electronic trading-partner agreement mark-up language. Previously, he has been involved in the architecture and design of the IBM Supplier Live solution for Ariba Buyer, IBM iSeries Connect, and IBM Commerce Integrator. Mr. Shaikh received an M.S. degree in computer engineering from Syracuse University, Syracuse, NY.

T. K. Sreeram *IBM Software Group, J2 B39, 17 Skyline Drive, Hawthorne, New York 10532 (electronic mail: sreeramt@us.ibm.com).* Mr. Sreeram is a development manager with the WebSphere Commerce development team. He received his master's degree in computer science in 1994 at the Kerala University, Kerala, India. His interests include catalog management, B2B gateways, and e-commerce.