DYNAMIC INTEROPERABILITY CONTRACT FOR WEB SERVICES

COPYRIGHT NOTICE

[0001] A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

REFERENCE TO COMPUTER PROGRAM LISTING APPENDIX

[0002] A computer program listing appendix appears immediately before the claims. The computer program listing appendix includes the following program excerpts:

InteroperabilityContract.xsd	(schema for overall contract.)
GeneralContract.XSD	(schema for general information.)
RoutingContract.XSD	(schema for routing of messages.)
TransformationContract.XSD	(schema for transformation of documents.)
SecurityContractKeyInfo.XSD	(schema for keys used for security.)
SecurityContract.XSD	(schema for security contract output from negotiation.)
InteroperabilityContract.XML	(example of interoperability contract.)
ComputeSecurityContract.XML	(computed security contract in example.)

BACKGROUND OF THE INVENTION

[0003] The present invention relates to machine-readable data structures and dynamic calculation of data structures to support interoperability. More particularly, it relates to aspects of data structures that enhance interoperability and dynamic generation of the data structures. Particular aspects of the present invention are described in the claims, specification and drawings.

[0004] Business-to-business (B2B) and application-to-application (A2A) electronic commerce are replacing former protocols for electronic data interchange (EDI). As businesses strive to improve their efficiency with B2B and A2A systems, a number of incompatible platforms and competing standards have emerged. Among compatible standards, gaps remain to be filled. For instance, the industry has defined what a simple web service is. Standards related

to simple Web service include UDDI, WSDL, XSDL and SOAP. However, these standards do not fully meet the security, reliability, manageability, and choreography requirements for practical B2B and A2A electronic commerce. Security in particular presents numerous options and configuration issues. Collaborative web services and their security needs are expected to evolve as non-web businesses do. There is no any comprehensive or unified device or method that dynamically resolves and updates security options and configurations as web services evolve.

[0005] There are a number of industry initiatives to extend standards applicable to B2B and A2A electronic commerce. Choreography efforts include ebXML/BPSS from OASIS, WSFL from IBM, and XLANG from Microsoft. Conversation efforts include ebXML/TRP from OASIS and Microsoft's WS-routing. Further information regarding ebXML initiatives is available at http://www.ebxml.org/specs/index.htm# whitepapers, where the article "Collaboration-Protocol Profile and Agreement Specification Version 1.0", by ebXML Trading-Partners Team (May 10, 2001) is found. Some information also is found in U.S. Pat. No. 6,148,290, for unambiguous rules of interaction and service contract enforcer logic. The dominant security effort is WS-security from IBM and Microsoft, there is also a complementary security effort in OASIS called SAML. For reliability, there are proposals from Microsoft, ebXML/TRP from OASIS, and HTTPR from IBM. W3C is addressing standardization in all of these areas. Key industry players have formed a rival consortium called WSI. However, they have not addressed the dynamic security negotiation issue.

[0006] In ebXML CPP and CPA, the parties interoperating define the profile called a CPP for their interoperation rules for their services in a single registry. The two profiles can be intersected to deduce the default interoperation agreement called a CPA. Alternatively the two parties agree on a specific set of interoperation rules between called a CPA. The problems with ebXML CPP and CPA include: They assume that the sending and receiving parties are in the same registry. The interoperation rules are insufficient to cover many aspects of interoperation. When used, they assume that a signed copy (signed by both parties) of the CPA is kept in a registry. This makes it cumbersome to maintain and modify. It is directly inconsistent with dynamically computing an interoperability agreement. Accordingly, instead of addressing dynamic computation with caching at runtime when a services invokes another service, but the talks about pre-downloading and local installation, which makes managing changes to the CPA difficult and not automatic.

[0007] Accordingly, an opportunity arises to develop methods and devices that dynamically determine interoperability agreements for trading partners.

SUMMARY OF THE INVENTION

[0008] The present invention relates to machine-readable data structures and dynamic calculation of data structures to support interoperability. More particularly, it relates to aspects of data structures that enhance interoperability and dynamic generation of the data structures. Particular aspects of the present invention are described in the claims, specification and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] Figure 1 illustrates communities and networks of communities, which are one environment in which machine-readable, dynamically negotiated interoperability contracts are useful.

[0010] Figure 2 illustrates multiple hub and spoke organizations that overlay the same connectors to support different transport/envelope protocols and technologies.

[0011] Figure 3 illustrates alternative embodiments for obtaining receiver's information when the sender is local to calculations of the security, transformation and other arrangements.

DETAILED DESCRIPTION

[0012] The following detailed description is made with reference to the figures. Preferred embodiments are described to illustrate the present invention, not to limit its scope, which is defined by the claims. Those of ordinary skill in the art will recognize a variety of equivalent variations on the description that follows.

[0013] Figure 1 illustrates communities and networks of communities, which are one environment in which machine-readable, dynamically negotiated interoperability contracts are useful. Among these communities, a community maintains a local registry that includes information such as users, companies, services and connectors that are part of the community. The community can be a marketplace, an enterprise or a sub enterprise. Communities can belong to one or more community networks. Typically, communities and networks have some common business interest. Interoperation is between member communities in one or more community networks. The networks include a gold marketplace network 1, a precious metal marketplace network 2, a private network 3 and a global trading web network 4. In this illustration, the gold marketplace network 1 and the precious metal marketplace network 2 are

contained within the global trading web network 4. The precious metals marketplace network 2 includes gold and silver marketplaces 14, 13. Gold marketplace customers can trade silver in the silver marketplace 13 and silver marketplace customers can trade in gold 14. One community, PQR Enterprise 17 belongs to the gold marketplace network 1, the private network 3 and the global trading web network 4; another community, ABC Big Supplier 18 belongs to the private network 3. In this illustration, XYZ Gold 14 is a marketplace or community for trading gold. Enterprises belong to this community. Enterprises like PQR Enterprise 17 that have formed a community by themselves belong to the gold marketplace network 1. These communities are part of the gold marketplace network 1, and the global trading web network 4. Small supplier 15 is part of the gold marketplace community. Other enterprises 16 are communities that are part of the gold marketplace community network 1. The connections between XYZ Gold 14 and other gold marketplace entities 15-17 indicate that the gold marketplace requires all traffic between enterprises (communities or otherwise) transacting gold trading to be routed through XYZ Gold14, for instance, to collect billing and business intelligence information. PQR Enterprise 17 is a community is part of the gold marketplace and also part of local private network with supplier 18. Small supplier 15 may be an individual small supplier that does not want to form a community by itself and instead registers its metadata, such as users, organizations, services and transformations, in the registry of the gold marketplace. On the other hand, ABC Big Supplier 18 has formed a private network of its own, for instance because it wants to keep its metadata, internal back office systems and transformations hidden from general public access because they were developed at considerable cost. Because PRQ 17 is a customer of ABC 18, it participates in the private network 3. Financial service provider DEF Financial 12 wants to provide financial services to anyone in the global trading web network 4, such forms a community of its own and registers with the global trading web root 11. A network of communities makes available a global registry of communities. The global registry permits lookup of the community and determination of one or more routes to that community, or to external connectors through which the electronic commerce documents bound for the community may be routed. Documents routed from one community to another may be routed directly between external connectors for the two communities or indirectly through one or more intermediary communities. Business and security rules for transactions involving the communities also can be defined and maintained in community registries. In general, figure 1 illustrates the mixed loyalties of entities and communities that create an impetus for interoperability among electronic commerce platforms.

[0014] Connector is a general term for applications that communicate with other applications. Connectors may communicate on a peer-to-peer (P2P) basis or on a directed basis

through other connectors that function as hubs, gateways, external ports, central connectors, etc. Connectors that communicate P2P are able to communicate with other connectors that use the same transport/envelope protocols. Connectors that communicate P2P optionally may enlist the assistance of other hub connectors that perform translation services, when trying to communicate with a connector that does not use the same transport/envelope protocol. Connectors that communicate on a directed basis communicate through hub connectors according to routing rules. Routing rules among connectors can be mapped in a directed graph, supporting one or more hub and spoke topologies for one or more transport/envelope protocols. A hub and spoke topology directs communications along spokes to hubs, in one or more tiers. This facilitates centralized services such as billing, business intelligence collection, tracking, auditing, accounting, or others. Multiple hub and spoke organizations may overlay the same connectors to support different transport/envelope protocols and technologies, as suggested by Figure 2. For instance, a stronger hub and spoke organization may be required to use Sonic as a transport technology than to use HTTP or HTTPS. Optionally, communication routes may depend on whether the source and destination are part of the same community. Within a sub-community (which may include the whole community), centralized functions may be unneeded and P2P communications permitted among connectors that otherwise are directed to communicate with parent connectors when communicating with destinations in other sub-communities.

[0015] Connectors may be labeled simple connectors (sometimes simply called connectors), hubs (sometimes called gateways or routers) or central connectors. Alternatively, they may be described functionally. Simple connectors are directed to communicate via hub connectors, except when they are permitted to communicate P2P among connectors in the same sub-community. So-called hubs are used by connectors that are explicitly directed or linked to them. Hubs may serve more than one function and, accordingly, may appear more than once in a route from a source to a destination. Hubs forward electronic commerce documents or messages. Hubs also may translate among transport protocols that support a common envelope protocol. For instance, a hub may translate envelope protocols and also implement a different transport protocol upon transmission than upon receipt. A central connector is a special case of a hub, which can be used by connectors that are not explicitly directed or linked to them. A central connector is useful, for instance, to carry out translation functions when traversing connectors from a source according to routing rules does not lead to any hub that supports the transport/envelope protocol used by the destination.

[0016] Aspects of the present invention address federated registries, components of an interoperability contract data structure, dynamic negotiation of the interoperability contract. The

scope of a registry is a community. A community can be an enterprise, a marketplace or a sub-enterprise in a larger distributed enterprise. The parties who interoperate might be in different communities. For example, one might be in a suppliers community and another might be in a buyers community. Therefore, a federated scheme for storing profiles and agreements should be used. For interoperation, the present invention goes beyond ebXML and other conventional approaches to e-commerce interoperation. An interoperability contract is extended to include combinations of the following: The route to follow in conveying messages between the services, conforming to defined routing rules. For example a rule might say that all messages to/from a web service should be fronted by a particular router. The route includes automatic routing through gateways for envelope transformations, such as between SOAP and EDI envelope protocols. The signing, authentication and encryption policy on a message part by message part basis is specified, as there can be multiple parts in a message, where the policy includes the algorithm, the technology (for example XML encrypt, SMIME, PKCS#7) and elements (for example an XML element in an XML document.) The transformation rules are specified for documents included in message parts, on a part-by-part basis. For example, if transformation for version interoperation is permitted and if so if the original should be attached. Specific transformation logic also can be identified. The version of the message exchange choreography to be used is identified. For example, a service might support multiple versions of a choreography, so services benefit from knowing the right version that the sender and receiver support. Certain message conveyance policies are set, such as whether to archive messages, to use reliable delivery, and to require a non repudiation receipt of acknowledgement. Differences in sent and received messages that need to be bridged are addressed by envelope adjustment or envelope transformation. For example different envelope extensions used, differences in message part order, different envelope protocols. The connectors in the route that serve various interoperation functions, consistent with on capabilities of the connectors, are registered in the registry. One of skill in the art will recognize that the preceding and following aspects of the present invention can be combined in many useful subsets; the invention is not intended to be limited to an interoperability contract that includes all aspects of the present invention.

[0017] The interoperability contract is normally derived by intersecting the policies and interfaces of the sender and receiver services. However, overrides are possible that indicate decision rules that should used to resolve conflicts between the sender and receiver. For example decision rules may determine that sender wins, receiver wins, most stringent policy wins, etc. This is useful for supporting service modifications, as the interoperation contract is computed and signed by a trusted service, such as a community root party who is trusted. The

interoperability contract may be dynamically computed when a service is invoked and may be locally cached at the message sending site.

[0018] When the invoking service is in one community and the invoked service is in another community, the contract calculation may be performed by a distributed logic that gets the send side and receive side information from community registry local to both services and intersects it. Any overrides is defined on the receive side and send side are noted and copied to the complementary side for approval, if there is no prior approval. At contract creation time, the resulting interoperability contracts, calculated on a distributed basis, end up being the same, as a cross community online negotiation process can be used to address the overrides. Much complexity is hidden from the service writer and the interoperation contract is automatically deduced from the registry. This greatly simplifies service development.

[0019] The interoperability contract should be dynamically computed to avoid major synchronization problems by installing the contracts in local machines. This does not necessarily require re-computation for each message, as a cache can supplement the dynamic computation. The cache could be kept coherent by invalidation notifications on changes in the registry or expiration policies. A cache keeper could subscribe to any required notifications. The interoperability contract can be dynamically computed upon initiation of a web service from the sending services connector (or a proxy for it that knows how to handle it). The contract may be attached to a message after computation, so that intermediate connectors between the communicating services understand their roles in the message exchange. For instance, the contract can specify which connector should perform version transformation, signing, encryption, etc.

[0020] Aspects of the present invention extend across multiple dimensions of interoperability. For true end-to-end message interoperation, there are many dimensions of interoperability to address. Addressing any one dimension of interoperability advances e-commerce using web-based services. Addressing combinations of interoperability issues can produce significant advances. In the discussion that follows, more than a dozen dimensions of interoperability and solutions within the scope of the present invention are presented.

1. Transport protocol interoperation

[0021] One dimension of interoperation is transport level interoperation. According to aspects of the present invention discussed more fully in the related applications, the allowed and supported transports are tied to the envelope protocol used. In contrast, in ebXML the allowed transport is HTTP(S). For MML, the allowed transport is Sonic. For Biztalk, the allowed

transport is HTTP(S). In one embodiment of the present invention, the allowed transports are HTTP(S) and sonic.

[0022] With sonic, the reliability is pushed down to the transport level. The sonic reliability protocol is an extremely good algorithm. Long term, HTTP(S)R, if standardized, may provide reliability at the HTTP(S) level itself. The solution used now by ebXML and Biztalk is reliability with a envelope protocol dependent approach over HTTP(S). SOAP, without extensions, provides no support for reliability at the envelope level.

[0023] The present invention includes support for negotiation of a transportation protocol among to supported protocols. In one embodiment, this involves a choice between HTTP(S) and sonic. As additional transports are adopted for e-commerce, the present invention can include those additional options in negotiations.

2. Envelope protocol interoperation

[0024] In one embodiment of the present invention, the envelope protocols supported are MML, C1 SOAP, email, and external SOAP, which allows any combination of optional extensions like C1 address, conversation and message info, manifest, SAML and SOAP with attachments. Services exposed with pure SOAP, SOAP WA, standard WSDL and discoverable with UDDI are called simple web services in the industry. However C1 SOAP, while inter-operating with endpoints that are simple web services (developed with third party development environments and third party execution environments) also supports native web services with reliability, security, and participation in bi-directional choreography. Back office systems exposed with J2EE CA or EJBs can be wrapped as a simple web service by third parties. This embodiment can interoperate with them, as well as supporting email protocols and external SOAP.

[0025] Supported protocol define allowed transport, reliability and security protocols. They also defines the way services and parties are addressed in that protocol and data linkages to tie related messages together. Message routing and dispatching are based on the address defined.

[0026] Envelope protocol determination and transformations can be supported by the interoperability contract. This is one of the ways in which the interoperability contract goes far beyond a typical ebXML CPA contract. Again, the interoperability contract may include information about the route to follow, the transformations to do and where to do them, things to be signed or encrypted ands where to do it and what algorithm to use, the name and version of the choreography, and the sending/receiving TP/service/service version/operation. The interoperability contract can be used to drives intermediate connectors along the route between

services. The segment of the route between participating services is the so-called "intelligent interoperable network," which adds value even if the endpoints strictly follow standards without using software developed by the assignee of this patent.

[0027] Interoperation between envelope protocols is through gateways. Different versions of the same protocol may be treated as different protocols. The router knows to transparently route a message through the appropriate set of gateways for interoperability. The dispatcher in the destination connector hands an inbound message to the appropriate component. This dispatching again is based on rules driven by the target address and other envelope fields.

[0028] One variation of envelope protocol interoperation is where we have a protocol with a baseline and multiple options that can be used. An example is external SOAP, with SOAP with attachments, routing, security, SAML etc. being optional. If the sender specifies one set of options and the receiver specifies another set, the point of entry into the network would compute if interoperation is possible and if so how. It would automatically add optional blocks based on rules and strip unwanted blocks at a point of exit from the network.

[0029] When we transfer XML data from a SOAP body to a document in a MIME part or vice versa, we could consider this to be a form of envelope interoperation. Such transformations occur when transforming between Biztalk and ebXML or SOAP and ebXML. It also occurs for SOAP-to-SOAP interoperation where the sender puts the payload in an attachment and the receiver expects it in the body (or vice versa).

3. Security protocol interoperation

[0030] One issue with envelope protocol interoperation is that the security protocol supported is defined by the envelope protocol and transforming between security protocols is near impossible. For example, switching from XML signature supported by envelope protocol A to PKCS#7 supported by envelope protocol B is not possible. If the receiving service requires the original signature or encryption for interoperation, the gateway should return an error to the sender, unless the gateway is trusted to transform security protocols. One approach to overcoming security protocol incompatibilities is to trust the gateway to verify the signature in the message and decrypt (the encryptor uses the gateways key) and resign and re-encrypt messages. A trust scheme is instituted, whereby the gateway's signature can be trusted by the receiver.

[0031] SOAP extensions proposed in the industry include WS-security (part of GXA). Embodiments of the present invention can support WS-security, including WS-security for C1 SOAP. At this time, such security extensions are optional and if a foreign web service has not

adopted WS-security, it could delegate to the point of entry into the interoperability network the authority to sign and encrypt messages on its behalf (the point of entry has access to the user key). This works if the point of entry into the network is located within the enterprise with the foreign web service.

[0032] No message should be accepted into the interoperable network without being authenticated first (unless the invoked service expressly does not care).

[0033] One aspect of security protocol interoperation is when the sender and receiver specify different security policies and capabilities. The interoperation framework has to compute if interoperation is possible and if so how.

4. Interoperation between different types of services

[0034] Typically, services are registered in the collaborative registry unless noted otherwise. In the context of this discussion, it is expected that a collaborative web service will interact at least one interface with another collaborative web service.

[0035] There are so-called simple, high performance and collaborative web services. There are also native and foreign web services. Lastly there are registered and unregistered services. A simple web service does not use signing, encryption, reliable messaging and does not require authentication from a central trusted party. It also does not support bi-directional choreographies. In other words, each invocation of a simple web service is independent of all previous invocations of the simple web service and there is no choreography context being kept in the simple web service, and no knowledge of return addresses in the context so it can reply back later. A high performance web service can include better reliability and security. A collaborative web service can be simple or high performance and in addition support bi-directional choreographies. Typically, web services other than those prepared by the assignee of this application (foreign web services) are simple web services.

[0036] As described throughout this application and the incorporated applications, aspects of the present invention can extend the mechanisms for e-commerce in numerous ways. Innovative web services can be registered in the collaborative registry as are high performance web services and collaborative web services. Support can be provided for a continuum between native simple and high performance web services where elements can be added one by one. A high performance web service can declares in the registry what elements it supports. It will be possible to download the WSDL definition of an innovative native simple web service (from UDDI or from Commerce One's own collaborative registry), which identifies a service port that is the URL of a point of entry into the network. Messages conveyed through the port of entry

will automatically be routed from there to their logical destinations. Messages routed in accordance with the present invention include or are governed by an interoperability contract that governs what happens at every hop. Native web service can invoke a native or foreign simple web service.

[0037] Foreign simple web services can be supported by an innovative network. If the foreign web service knows the innovative addressing and message identity and correlation SOAP extensions, it could even participate in bi-directional choreography as a collaborative web service. Foreign web services may use a combination of innovative SOAP extensions. They do not need to access a community registry or understand an interoperability contract. The present invention could be extended to provide software to build foreign web services and third party software should be used. Foreign web services can be invoked by any native web service or any other foreign web service through our network. Foreign web services can use external SOAP or email. In the case of email, a human user using an email browser could "implement" the web service and interoperate with both simple and collaborative native or foreign web services. The WSDL definition of the foreign web service can be downloaded from the collaborative registry or from UDDI. A foreign web service invokes a web service in an innovative network by invoking a URL at a point of entry into the network. A collaborative foreign web service is provided the URL of the point of entry into our network in a SOAP extension as part of invocation by a native collaborative web service, so it can dynamically respond back later if it understands the SOAP extension.

5. Network and location independent interoperation

[0038] The location of the destination services component should not matter and the marketplace or enterprise community that the service is registered in should not matter. The routing algorithm should transparently handle location transparency and marketplace or enterprise community transparency. Routing along with the transport and security mechanism should support automatic tunneling through appropriate enterprise and marketplace firewalls without compromising security.

6. Platform independent interoperation

[0039] A platform may include the hardware/operating system the software runs in and the development and execution environment of the server the business service runs in. It also may involve the server technology (J2EE app server, web server, servelet runner) the software runs in. The hardware part of independence can be achieved by using 100% pure Java. The independence from development/execution environment can be achieved by supporting strict

standards based wire level interoperation with foreign connectors and servers. The server technology independence can be achieved by making components embeddable and conforming to J2EE standards.

[0040] When vendor supplied components are platform independent, a customer can develop services using their preferred development/execution environment from their preferred favorite vendor and accessed with their favorite client side tool. Such services can still interoperate with vendor developed services with interoperation value added by the intelligent network, and all services can be composed into more complex services with composition capabilities using a process flow engine.

[0041] A light weight commerce web services server can be deployed based primarily on message interoperation components. A lightweight server would be targeted for supplier connectivity, gateway writers and for the ISV market. A more complete embodiment of a collaborative web services server that is a superset of the lightweight edition. T he lightweight edition includes basic development tools for document related development, but primarily leverages third party tools for service development. A sophisticated full development environment for UI and document based process centric self-contained or composed services may be included with the collaborative web services server embodiment.

7. Back Office system interoperation

[0042] One aspect of interoperation with foreign connectors is interoperation with back office systems. Aspects of the present invention allow back office systems to be exposed so that the look just like a plurality of services from a messaging level and from a discovery level. Toolkits will allow back office system operators to expose their interfaces as simple web services, or wrap their custom adapters as a web service. Custom integration brokers will be able to integrate established EAI technologies with the innovative messaging system or to directly construct a web services interface. Another embodiment of integration with a back office system is email support. An email server can be used to integrate a back office system with the innovative network.

[0043] Exposing back office systems as web services could involve specialized transformation schemes not based on XML. Examples are transforming between DB and XML or XML and flat files, or transforming between J2EE CA 1.0 record structures and XML. All this is hidden from downstream web services and transparent to downstream web service developers.

8. Service discovery and cross community interoperation

[0044] In the future, it is likely that interoperation between trading partners will become more dynamic. A discovery mechanism is a useful to find a trading partner to do business with, before setting up the business relationship. Discovery of services and trading partners offering them is done through the UDDI standard. A more powerful tool that UDDI supports is invoking innovative registry web services. Inventions related to the present invention will provide support for uploading data to a public UDDI registry or to a private UDDI registry that serves as yellow pages for a community or a set of communities. Discovery across the network of communities is possible.

[0045] For discovery across communities, each community may have a list of global white page communities or global yellow page registries associated with it. Global white page communities contain transport addresses for routing a request into a set of advertised communities. Global yellow page registries contain the trading partners and services of a set of advertised communities along with aliases and categories. Searches are done by categories. Since interoperation is bi-directional, two communities can subscribe to a common global white page community or have routing information to each other directly within their community registries. Two communities can discover objects in each other if they subscribe to a common yellow page registry. Typically, a yellow pages registry is hosted within a white page community.

[0046] Programming registry access interfaces are supported for not only discovery, but also trading partner information including roles and privileges, and users and organizations and their relationships. Also there is support for getting the technical information for interoperation including WSDL files, service interfaces, transformation code and schema files.

9. Registry version interoperation

[0047] Registry services may be configured as other web services and benefit from the interoperability support of all services.

10. Document semantic interoperation

[0048] The infrastructure does not care about the semantics of the payload. However document semantic interoperation is what allows services using differing document to enjoy end-to-end interoperation. The sender and receiver have to agree to the document semantics, such as document family members and transformation among the members, to facilitate interoperation. For interoperation with back office systems, document standards may include Idocs and OAGI.

11. Document Version interoperation

[0049] The interface of a receiving operation in a service can define support for one or more versions of a document. The innovative version interoperation system transforms between the sent document and the expected document to be received and tries to reduce loss by picking the best-received version. The transformation occurs before the message is signed and encrypted on the send side.

[0050] The registry supports major and minor versions within document families. Major versions may conform to different schema languages. Minor versions are expected to add optional parts to a base version.

12. Schema language interoperation

[0051] The schema languages for payload XML documents are defined by the envelope protocol. Examples of schema languages are SOX and XSDL. These are languages to describe the schema of an XML document. An XML instance of a schema in one language is different than an XML instance of an equivalent schema in another language. Therefore, schema language instance transformations should be supported by transformations in gateways.

[0052] Gateways may perform so-called syntactic transformations where the structure of the payload (relationship of elements) and semantics is not changed but the syntax and packaging is changed. A compatible structure is converted to an exact equivalent XML markup and vice versa.

13. Dependency on location and order of interoperation steps

[0053] From this discussion, those skilled in the art will see that the interoperability contract is one way of assuring that interoperation steps are carried out at agreed locations and in an agreed order. A message from sender to receiver travels through a series of connectors where different connectors do various steps for interoperation. There is interplay between the location and order of schema language instance transformation, version transformation, envelope transformation, signing, and encryption. The infrastructure properly orders the transformations.

14. Service version interoperation

[0054] Web services are defined by how they appear outside, in terms of their registry description and when addressing messages to them. It will be natural for a service to be upgraded and a service version to change over time. A new version of a service might have added operations or added or deleted optional parts in an existing message. It might also have

changed the set of choreographies supported and the location of a part in the message. Choreography interoperation described can be used to allow senders to know if they should invoke the new operation. In addition, the version numbers of the services are made known to the sender and receiver, so they can respond appropriately.

[0055] The infrastructure takes care of interoperation when the set of optional parts are different or when a body part becomes an attachment or vice versa.

15. Choreography interoperation

[0056] There are at least two embodiments that support choreography. One embodiment defines a process flow and has all participants run their messages through this process. The process flow runs in a process flow engine in a service. Another embodiment supports direct messaging between the endpoint services with knowledge of the choreography details in the endpoint service themselves. A process flow engine process sends and receives messages with other services and therefore can be made to look like a service itself. This abstraction can be very useful.

[0057] Process flow engine processes should appear as a service. The applications that want to interact with the process send to and receive messages from this service. Because of this abstraction, the process flow engine process can also be used to compose a bigger service by using the process definition to tie together a set of services into a flow and expose the larger service. Moreover, a process flow engine process engine can be made available in every innovative service and therefore distributed processes can be built that span across multiple process engines. This is possible because each sub process in the distributed process looks just like a service and a sub process invoking another sub process is treated just like a service invoking another service. The various sub processes interact with messages and the messages could carry process flow context available in each sub process to more tightly integrate the sub processes.

[0058] A consideration in bi-directional choreographs between services is the ability to know the sending TP/service/operation, particularly when one of the services does not directly support choreography or conversation ID extensions. A method to correlate related messages with conversation ID is useful. It is possible to have a virtual conversation with a simple web service, which does not support choreography by using payload data to correlate related messages that form the conversation. A process flow engine includes logic and resources to perform the correlation. For messages from foreign connectors without the addressing extension (typically true with back office systems) the message could be sent to a fixed service that looks

at the payload, the registry or a local database to deduce the destination address before forwarding the message on. This capability is called logical routing and process flow engine facilitates this, based on a configured specification of fields in the payload to examine, from which the conversation ID can be inferred.

16. Choreography variation interoperation.

[0059] Choreography ties a set of service types offered by the participants together. All variations of a choreography form a family where the first message are substantially the same. There should be only one family supported between two services that interoperate and the choreographies in that family could be ordered by preference. However a service might support multiple families of choreographies involving different combinations of services. Choreographies can be multipolar.

[0060] In one embodiment of choreography negotiation, when the first message in the choreography is sent, the sender and the receiver are told the choreography variation picked by the system. The choreography between these can then not change. They then adjust their processing accordingly. If a new service is added to the conversation, the sending service may chose between acting as a bridge between choreography variations supported by different services in a multipolar choreography or forcing use of the selected choreography. Choreography negotiation is further described in one of incorporated, commonly owned applications.

17. Hiding the complexities of interoperation

[0061] Services that interact in accordance with aspects of the present invention may need to know little or nothing about interoperation, as the complex issues can be taken care of under the covers. New modules to implement interoperation can be configured. These modules take care of the complex issues related to interoperation driven by registry metadata. API abstractions can be provided to hide the envelope structure completely and hide as much as possible of the envelope specific field semantics and syntax. All the security policies can be included in the interoperability contract, simplifying the service developer's efforts to implement applications.

18. Mechanisms to Limit interoperation

[0062] One barrier to true interoperation is security. The model is that the infrastructure authenticates the sender and the service authorizes it possibly based on metadata captured by the registry. The barriers include business rules, subscriptions and hidden services. Business rules

sometimes should limit interoperation across communities or within a community. Subscriptions may be required before interoperation, as indicated by the provider's service policy. It also is useful to have hidden services that are not visible outside the community or are only visible to specific parties.

[0063] Figure 2 illustrates the usefulness of a dynamically negotiated interoperability contract between a producer service and a consumer service. The principal features of the figure include a registry 201, a web services engine 202 including logic to dynamically determine an interoperability contract, a producer service 203 that exposes a choreographed interface to an internal process flow 204, and a consumer service 205. The figure text indicates that this example involves an order receiving system that produces order acceptances. The producer and consumer services have their own capabilities and policies for choreography, service version, documents, security authentication, security encryption, security signing, envelope protocol and transport 213, 215. A dynamically negotiated interoperability contract reduces the extent of pair wise configuration required to set up or maintain a web of services. It provides unambiguous rules for resolving differences between policies set by participants. As the participating services evolve, the dynamically negotiated interoperability contract evolves too.

[0064] Dynamic negotiation of an interoperability contract presents a remarkable deviation from conventional approaches that more nearly approximate legal contract negotiation. Dynamic negotiation begins from a producer service's description of its availability, capabilities and policies. A consumer service can readily discover the producer service using a discovery protocol such as UDDI. The producer and consumer have machine-readable specifications of their capabilities and policies. One or more schemas recognized by the producer and consumer unambiguously defines how the respective parties capabilities and policies are to be interpreted and intersections found. Instead of inviting negotiation to resolve differing interoperability terms, the system provides decision rules regarding how to resolve two types of conflicts: conflicts between preferences for alternative options and conflicts regarding whether to apply security measures such as signing and encryption to particular parts of messages that will be exchanged according to the dynamically negotiated interoperability contracts. The decision rules for preferences may be standard rules, such as receiver wins, sender wins, most stringent requirement wins, least stringent requirement wins or a weighted consideration of both parties' preferences is applied. The decision rules for whether to apply security measures, for instance, are similar. These decision rules, including overrides, are further discussed in the Dynamic Negotiation Of Security Arrangements Between Web Services patent application filed concurrently with this application and incorporated by reference. In some instances, the

producer may require subscriptions before consumers can interact with the producer. This may facilitate credit and authentication checks and the like. The framework of intersections and decision rules allows a trusted software agent to dynamically negotiate an interoperability agreement, especially if a subscription has been accepted by a producer. This use of a trusted software agent authorized to dynamically negotiate an interoperability contract is a remarkable departure from the more conventional CPA-styled interoperability agreement that is cryptographically signed by both producer and consumer before it can take effect. (While this description is stated in terms of producer and consumer services, to assist the reader's understanding, it applies equally to two or more services, irrespective of their roles as producer, consumer, intermediary or otherwise.)

[0065] A set of schemas and sample interoperability contract provide additional detail regarding aspects of the present invention.

[0066] The schema Interoperability.XSD, in the source code appendix, can be used to model an interoperability contract, including several aspects of the present invention. In this embodiment, the machine-readable output files is an XML document. In other embodiments, other data structures may be used to store the same information, for instance a tree structure modeled after the XML code. The schema Interoperability.XSD is best understood by loading the file into an integrated development environment (IDE) such as XML Spy TM, which provides several alternative views of the schema, including a documentation generation view. Viewed in Spy's schema design view, Interoperability.XSD components include a general contract section, a routing contract section, a transformation contract section, a security contract section and a contract signature. The four sections each incorporate by reference another schema, which is discussed below. The contract signature, unlike conventional interoperability contracts, is applied by a software agent trusted to negotiate the contract. Separate signatures of the parties to the contract are not required. Parts of the contact signature includes the SignedInfoType, the SignaureValue, KeyInfo and the ObjectType, as further documented in the source code.

[0067] The schema GeneralContract.XSD, also in the source code appendix, can be used to model the general section of an interoperability contract, including several aspects of the present invention. GeneralContract.XSD components include to and from information, ErrorHandling, and DeliveryReceiptHandling. The components optionally include RequiredMessageParts and OptionalMessageParts, and sending and receiving connector capabilities. The to and from information relates to the party / service / activities involved. The error-handling component describes capabilities and optionally identifies where to send error

messages. Like ErrorHandling, DeliveryReceiptHandling is a capability parameter with an optional address for messages. Delivery receipts are used to implement non-repudiation. The required message and optional parts are as named. The role of required and optional parts in service versioning and document family versioning is more fully discussed in the incorporated applications. The sending and receiving connector capabilities list the attributes of the connectors and the values of the attributes (such as capable of signing or encryption.) The capabilities are optional, because they may not appear for non-collaborative requests or for one-way messages. These components are further documented in the source code.

[0068] The schema RoutingContract.XSD, also in the source code appendix, can be used to model the routing section of an interoperability contract, including several aspects of the present invention. Viewed in Spy's schema design view, RoutingContract.XSD components specify a route. A Route includes two or more RouteNodes in the route, including the sender and receiver. Entry and exit channels to nodes are defined by the transport and envelope protocol used to reach or when exiting from a node. The symmetry of this information allows the exit and entry channels to reverse roles for a reversed route. This schema is further documented in the source code. Routing is more fully discussed in the incorporated applications.

[0069] As addressed in one of the concurrently filed applications, negotiation of security arrangements is carried out by a computer-based process that uses security profiles of sending and receiving services to determine a mutually agreeable security arrangement. Preferably, this security arrangement is negotiated or potentially updated regularly, without user intervention. This arrangement may be negotiated, updated or checked for validity at a user request or without user intervention whenever messages are exchanged or on some other periodic or occasional basis, such as monthly, weekly, daily, on occurrence of an event that impacts exchange of messages between a particular sender and receiver (e.g., a software component failure or a change in security preferences), when a previously negotiated arrangement fails, or on some other periodic or occasional basis. The schema SecurityContract.XSD, in the source code appendix, can be used as a model for preparing a machine-readable security interoperability contract document. In this embodiment, the machine-readable document is an XML document. In other embodiments, other data structures may be used to store the same information, for instance a tree structure modeled after the XML code. This schema defines policies and channels for security policies. A security channel defines resources and routes to resources that carry out security algorithms, such as signature, encryption and authentication algorithms. It also may include non-repudiation and authorization resources.

[0070] A set of computed security arrangements are partially reproduced below:

```
<SecurityContractICD ... >
  <SecurityPolicies>
    <SignaturePolicies>
      <XMLDsigPolicy PolicyId="P-XMLSignatureRSA-MD5-C14N">
      <SignaturePolicyAlgorithm>...</SignaturePolicyAlgorithm>
        <SignatureAlg...>MD5withRSA</SignatureAlg...>
        <HashFunction>MD5</HashFunction>
        <Canonical ...>...14n-20001026</Canonical ...>
        <Transform>...#RoutingSignatureT...</Transform>
      </XMLDsigPolicy>
    </SignaturePolicies>
    <EncryptionPolicies>
      <XMLEncryptionPolicy PolicyId="P-XMLEncrypt3DES-RSA-2048">
  <EncryptionPolicyAlgorithm>http://www.w3.org/2001/04/xmlenc#</EncryptionPolicyAlgorithm
>
        <EncryptionMethod>http://www.w3.org/2001/04/xmlenc#3des-
cbc</EncryptionMethod>
        <KeySize>2048</KeySize>
  <KeyEncryptionMethod>http://www.w3.org/2001/04/xmlenc#rsa-
1_5</KeyEncryptionMethod>
      </XMLEncryptionPolicy>
    </EncryptionPolicies>
    <EncryptionKeyInfo KeyOwner="x-
ccns:commerceone.com:CollaborationParty::sellParty">
      <PublicKeyID>DefaultTestCert</PublicKeyID>
      <X509Data>          <X509Certificate>LS0tLS1... ==</X509Certificate>
      </X509Data>
    </EncryptionKeyInfo>
  </SecurityPolicies>
  <SecurityChannel channelId="CHANNEL1" sourceConnector="x-
ccns:cup.commerceone.com:connector::centerSell" targetConnector="x-
ccns:cup.commerceone.com:connector::centerSell">
    <Confidential AlgorithmId="P-XMLEncrypt3DES-RSA-2048">
      <PublicKeyName KeyOwner="x-ccns:commerceone.com:CollaborationParty:
:sellParty">DefaultTestCert</PublicKeyName>
      <MessagePart PartName="Order" isOptional="false"/>
      <MessagePart PartName="Image" isOptional="false"/>
```

```
    </Confidential>
  </SecurityChannel>
  <SecurityChannel channelId="CHANNEL2" sourceConnector="x-
ccns:cup.commerceone.com:connector::buy" targetConnector="x-
ccns:cup.commerceone.com:connector::sell">
    <Integrity AlgorithmId="P-XMLSignatureRSA-MD5-C14N">
      <PublicKeyName KeyOwner="OwnerA">BuyerPublicKey</PublicKeyName>
      <MessagePart PartName="Order" isOptional="false"/>
    </Integrity>
  </SecurityChannel>
</SecurityContractICD>
```

[0071] This set of security arrangements has two major sections for security policy and security channels. In this example, there is one security policy applicable to the entire message and multiple security channels to implement parts of the security policy. The security policy section sets out the signature policy, and encryption policy and encryption key information. It also may set out policies regarding authentication, authorization and non-repudiation of sending or receipt. In this embodiment, the same signature and encryption policy is applied to all parts of the document. In other embodiments, multiple algorithms could be applied to different parts or different elements within a part. The algorithm selected for signature, encryption and authentication are abstracted through templates containing options sets, simplifying the selection of algorithms. Selected algorithms are associated with logic and resources, so different services or processes can be used for signing/verifying and encrypting/decrypting different parts of a message. A public key or certificate can be transmitted in the encryption key element of the security policy section. The security channel section describes services or connectors involved in applying security policies. For a particular policy, the channel section identifies a source connector that requires assistance in applying a security policy (e.g., the sending service requesting encryption), and a target connector that applies the security policy or acts as an intermediary to logic and resources that apply the security policy. For a particular security policy, such as signing, encryption, authentication, authorization or non-repudiation, specific information required to carry out the security policy is provided in the security channel section.

[0072] Figure 3 illustrates alternative embodiments for obtaining receiver's information when the sender is local to calculations of the security, transformation and other arrangements. In the figure, local 331 and remote 332 registries are indicated. In this example, the sender is local and the receiver remote. The sender's data is current and complete in the local registry 331. The sender's information is collected 321 and made available to the logic and resources

that compute the security arrangements 311. The receiver's data may be current and complete, for instance if the receiver is in the same community as the sender and there is a community-wide registry, or if the receiver's information has been recently obtained and locally cached. Depending on where the receiver's information can be found, 331 or 332, a process 322 or 323 is invoked to collect the receiver information and make it available to the logic that computes security arrangements. A set of security arrangements 301 result.

[0073] Two types of preferences may need to be reconciled. Both community and service-specific preferences may be stated. One type of preferences is among algorithm templates. A decision rule for choosing between options B and D might take into account one or both of the messaging services' preferences. For instance, the receiving service's preference (D) for signature or the sending service's preference (B) for encryption might be selected from among the matches. Taking both preferences into account, the most stringent (B) or the least stringent (D) might be selected. In another embodiment, the respective services might weight or score their preferences and a combined weighting or score may be used to take into account both preferences. The second type of preferences is for whether or not to sign or encrypt a part of a message. Decision tables may be used to implement the type of preference reconciliation related to whether to sign or encrypt part of a message. Again, decisions could be biased to accept preference not to sign or to accept the receiver's preference, or just the opposite. Some decision tables that could be used to implement possible decision rules follow:

		Sender Preference		
		Signature Required	Signature Optional	No Signature
Receiver Preference	Signature Required	Sign	Sign	Error
	Signature Optional	Sign	Don't Sign	Don't Sign
	No Signature	Error	Don't Sign	Don't Sign

		Sender		
		Encryption Required	Encryption Optional	No Encryption
Receiver	Encryption Required	Encrypt	Encrypt	Error
	Encryption Optional	Encrypt	Don't Encrypt	Don't Encrypt
	No Encryption	Error	Don't Encrypt	Don't Encrypt

	Sender

		Signature Required	Signature Optional	No Signature
Receiver	Signature Required	Sign	Sign	Sign
	Signature Optional	Sign	Don't Sign	Don't Sign
	No Signature	Don't Sign	Don't Sign	Don't Sign

		Sender		
		Encryption Required	Encryption Optional	No Encryption
Receiver	Encryption Required	Encrypt	Encrypt	Encrypt
	Encryption Optional	Encrypt	Don't Encrypt	Don't Encrypt
	No Encryption	Don't Encrypt	Don't Encrypt	Don't Encrypt

[0074] These formats for security decision rules apply with equal force to other preference negotiations. In some special cases, such as transformation, metrics of information loss or transformation accuracy may be applied, as described in the incorporated applications.

[0075] The schema TransformationContract.XSD, also in the source code appendix, can be used to model the document transformation section of an interoperability contract, including several aspects of the present invention. Viewed in Spy's schema design view, TransformationContract.XSD components specify one or more documents to transform and optionally specify response documents. DocumentToTransformType includes a source document ID and part name, and a receiver attachment preference flag. It optionally includes an attachment part ID and one or more transformation maps, that describe how to implement a transformation. This schema and particularly the transformation maps are further documented in the source code. Document transformation is more fully discussed in the incorporated applications.

[0076] A partial example of a computed interoperability contract is provided in InteroperabilityContract.XML, in the source code appendix. This example includes general, routing and transformation contract sections. See above for an example of a security contract section. The example is largely self-explanatory to those of skill in the art, particularly with the accompanying schemas available. Some highlights follow. The general contract section identifies this as contract as governing a collaborative interaction. Messages are archived for non-repudiation, error handling and other uses. Utilities are allowed to consider messages governed by this contract in compiling aggregate (or, configurably, specific) business intelligence information. A from address is given for a buyParty ConsumerOrderManagement

sendOrder activity. A historical DDID number or address further identifies the sending service. A receiving address is given for sellParty providerOrderManagement process order activity. The sender accepts asynchronous error messages using a C1 SOAP 1.0 envelop protocol to a specified address. The sender requires a delivery receipt, which the receiver can generate asynchronously. The required message parts or documents are Order and Image. Optionally, a someXMLPart can be included. Sending and receiving connector capabilities are enumerated for signing, encryption, archiving, message envelopes, manifest types, and delivery receipt types. A sample general contract section is part of the example in the source code appendix.

[0077] In addition to the general contract section, there are a routing contract section and a transformation contact section. The sample routing contract section follows:

```
<RoutingContract>
    <route:RouteNode preICDComputation="false" connector="x-
gtw:cup.commerceone.com:connector::default" isNative="true" connectorFunction="service-send">
        <route:EntryChannel envelopeProtocol="C1 SOAP 1.0"
transportSupportedMessageType="both"
transportPhysicalAddress="icdtest.commerceone.com::SOAP_buyspicenutmeg"
transportProtocol="SONIC" transportNative="true" transportReliable="true"/>
        <route:ExitChannel envelopeProtocol="C1 SOAP 1.0"
transportSupportedMessageType="both"
transportPhysicalAddress="icdtest.commerceone.com::SOAP_buyspicenutmeg"
transportProtocol="SONIC" transportNative="true" transportReliable="true"/>
    </route:RouteNode>
    <route:RouteNode preICDComputation="false" connector="x-
gtw:cup.commerceone.com:connector::default" isNative="true" connectorFunction="service-receive">
        <route:EntryChannel envelopeProtocol="C1 SOAP 1.0"
transportSupportedMessageType="both"
transportPhysicalAddress="icdtest.commerceone.com::SOAP_buyspicenutmeg"
transportProtocol="SONIC" transportNative="true" transportReliable="true"/>
        <route:ExitChannel envelopeProtocol="C1 SOAP 1.0"
transportSupportedMessageType="both"
transportPhysicalAddress="icdtest.commerceone.com::SOAP_buyspicenutmeg"
transportProtocol="SONIC" transportNative="true" transportReliable="true"/>
    </route:RouteNode>
</RoutingContract>
```

[0078] This sample illustrates application of the schema described above. Similarly, the sample tranformation contract, illustrating application of the transformation schema, follows:

```
<TransformationContract>
   <xform:DocumentToTransform>
   <xform:SourceDocID>publicid:com.commerceone.schemas:PurchaseOrder:3.5</xform:Sour
ceDocID>
      <xform:PartName>PurchaseOrder</xform:PartName>
      <xform:Attachment>false</xform:Attachment>
      <xform:TransformationMap>
        <xform:Connector>x-gtw::lion-z-
01.lion.commerceone.com::connector::buyspicenutmeg</xform:Connector>
          <xform:StartDoc>
    <xform:DocURI>publicid:com.commerceone.schemas:PurchaseOrder:3.5</xform:DocURI>
             <xform:DocName>PurchaseOrder</xform:DocName>
      <xform:Namespace>publicid:com.commerceone.schemas</xform:Namespace>
             <xform:Version>3.5</xform:Version>
          </xform:StartDoc>
          <xform:EndDoc>
    <xform:DocURI>publicid:com.commerceone.schemas:PurchaseOrder:4.0</xform:DocURI>
             <xform:DocName>PurchaseOrder</xform:DocName>
    <xform:Namespace>publicid:com.commerceone.schemas</xform:Namespace>
             <xform:Version>4.0</xform:Version>
          </xform:EndDoc>
          <xform:CommunityID>exostar</xform:CommunityID>
          <xform:TransformationMapURI>urn:x-
commerceone:transformation:1</xform:TransformationMapURI>
      </xform:TransformationMap>
   </xform:DocumentToTransform>
</TransformationContract>
```

[0079] From the preceding description, it will be apparent to those of skill in the art that a wide variety of systems and methods can be constructed from aspects and components of the present invention. One embodiment is a machine-readable data structure that specifies interoperability data. An environment in which this machine-readable data structure is useful is for interoperation between a consuming service and a providing or producing service. These services exchange documents via a network, optionally using intermediate connectors. The machine-readable data structure may combine any two or more of the following useful data elements: a route between the services, specified by the names of the services and the

intermediate connectors; a choreography version to be used for an exchange of messages; policies for archiving the messages, for assuring reliable delivery of the messages and for requiring a receipt acknowledgment; a specification assigning requirements for parts of a particular message and at least one signing algorithm to use; a specification of encryption requirements for parts of a particular message and at least one encryption algorithm to use; a specification of one or more authentication procedures to use; a specification of one or more transformation logics to apply to documents included in a particular message; and a specification of whether untransformed copies of the documents should be included with transformed copies the documents. The combinations specified in the accompanying claims are not meant to be exclusive. The permutations of two or more of the above useful data elements are hereby expressly described.

[0080] A further embodiment of the present invention is a machine-readable data structure that specifies current interoperability data prepared by a process. An environment in which this machine-readable data structure is useful is interoperation between a consuming service and a providing or producing service. The services exchange documents via a network. The services may optionally use intermediate connectors. Unlike static interoperation contracts, such as contracts that are signed by both parties, this machine-readable data structure is created by a process responsive to a request to initiate an exchange messages between the services. The processing in clues accessing interoperability data for the services, intersecting the interoperability data for the services, and, for intersections interoperability data that produce more than one mutually acceptable option, applying decision rules to select one option. This machine-readable data structure may include any permutations of useful data elements described in the prior embodiment. The decision rules used may be subscribed to by the services that are exchanging messages or may be adopted by subscription of the services to a trading community. Any of the decision rules described throughout this application may be used as a further aspect of this embodiment.

[0081] Another embodiment of the present invention is a machine-readable data structure that specifies one or more security channels. An environment in which this machine-readable data structure is useful is interoperation between a consuming service and a providing or producing service. The services exchange documents via a network. The services may optionally use intermediate connectors. The security channels apply to one or more of assigning, encryption or authentication. They also may be applied to authorization or to non repudiation, or any combination of these security-related tasks. The security channels themselves include specification of a connector originating a security-related request and a connector responding to

the security-related request, and a specification of the security related request. The security-related request may include one or more of the above listed security-related tasks. This data structure including security channels may be formed responsive to request to an initiate an exchange of messages between the services.

[0082] While the present invention is disclosed by reference to the preferred embodiments and examples detailed above, it is understood that these examples are intended in an illustrative rather than in a limiting sense. Computer-assisted processing is implicated in the described embodiments. Accordingly, the present invention may be embodied in methods for computer-assisted processing, systems including logic to implement the methods, media impressed with logic to carry out the methods, data streams impressed with logic to carry out the methods, or computer-accessible processing services. It is contemplated that modifications and combinations will readily occur to those skilled in the art, which modifications and combinations will be within the spirit of the invention and the scope of the following claims.

[0083] We claim as appears after the appendix:

COMPUTER PROGRAM LISTING APPENDIX

InteroperabilityContract.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
edited with XML Spy v4.3 U (http://www.xmlspy.com) by Rashmi Murthy (Commerce One)
  -->
- <xs:schema
    targetNamespace="publicid:com.commerceone:schemas/soapextension/contract/v1_0/InteroperabilityContract.xsd"
    xmlns:security="publicid:com.commerceone:schemas/soapextension/contract/security/v1_0/SecurityContract.xsd"
    xmlns:xform="publicid:com.commerceone:schemas/soapextension/contract/transformation/v1_0/TransformationContract.xsd"
    xmlns:route="publicid:com.commerceone:schemas/soapextension/contract/routing/v1_0/RoutingContract.xsd"
    xmlns:general="publicid:com.commerceone:schemas/soapextension/contract/general/v1_0/GeneralContract.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:icd="publicid:com.commerceone:schemas/soapextension/contract/v1_0/InteroperabilityContract.xsd" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <xs:import
      namespace="publicid:com.commerceone:schemas/soapextension/contract/general/v1_0/GeneralContract.xsd"
      schemaLocation="http://schemas.commerceone.com/schemas/soapextension/contract/general/v1_0/GeneralContract.xsd" />
    <xs:import
      namespace="publicid:com.commerceone:schemas/soapextension/contract/routing/v1_0/RoutingContract.xsd"
      schemaLocation="http://schemas.commerceone.com/schemas/soapextension/contract/routing/v1_0/RoutingContract.xsd" />
    <xs:import
      namespace="publicid:com.commerceone:schemas/soapextension/contract/transformation/v1_0/TransformationContract.xsd"
      schemaLocation="http://schemas.commerceone.com/schemas/soapextension/contract/transformation/v1_0/TransformationContract.xsd" />
    <xs:import
      namespace="publicid:com.commerceone:schemas/soapextension/contract/security/v1_0/SecurityContract.xsd"
      schemaLocation="http://schemas.commerceone.com/schemas/soapextension/contract/security/v1_0/SecurityContract.xsd" />
    <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd" />
  - <xs:element name="InteroperabilityContract">
    - <xs:annotation>
        <xs:documentation>Container for ICD blocks</xs:documentation>
      </xs:annotation>
    - <xs:complexType>
      - <xs:sequence>
```

```
- <xs:element name="GeneralContract"
    type="general:GeneralContractType">
  - <xs:annotation>
      <xs:documentation>General contract sub-block of
        ICD. This contains all general contract
        information</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="RoutingContract"
    type="route:RouteType">
  - <xs:annotation>
      <xs:documentation>Routing contract sub-block of
        ICD. Contains the end-to-end
        route</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="TransformationContract"
    type="xform:TransformationContractType"
    minOccurs="0">
  - <xs:annotation>
      <xs:documentation>Transformation contract sub-
        block of ICD. Contains transformation
        information required for version
        interoperability</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="SecurityContract"
    type="security:SecurityContractType" minOccurs="0">
  - <xs:annotation>
      <xs:documentation>Security contract sub-block.
        Contains security information needed to satisfy
        security constraints between the sending and
        receiving parties</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="ContractSignature"
    type="ds:SignatureType" minOccurs="0">
  - <xs:annotation>
      <xs:documentation>Signature for this
        contract</xs:documentation>
    </xs:annotation>
  </xs:element>
 </xs:sequence>
 <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
</xs:element>
</xs:schema>
```

GeneralContract.XSD

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
edited with XML Spy v4.3 U (http://www.xmlspy.com) by Rashmi Murthy (Commerce One)
-->
- <xs:schema
    targetNamespace="publicid:com.commerceone:schemas/soapextension/contract/general/v1_0/GeneralContract.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:gen="publicid:com.commerceone:schemas/soapextension/contract/general/v1_0/GeneralContract.xsd" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  - <xs:element name="GeneralContract" type="gen:GeneralContractType">
    - <xs:annotation>
        <xs:documentation>General information of the
          InteroperabilityContract</xs:documentation>
      </xs:annotation>
    </xs:element>
  - <xs:complexType name="ServiceActivityType">
    - <xs:annotation>
        <xs:documentation>Description of service and
          activity</xs:documentation>
      </xs:annotation>
    - <xs:sequence>
      - <xs:element name="Service">
        - <xs:annotation>
            <xs:documentation>URI of the service
              definition</xs:documentation>
          </xs:annotation>
        - <xs:complexType>
          - <xs:simpleContent>
            - <xs:extension base="xs:anyURI">
                <xs:attribute name="Version" type="xs:string"
                  use="optional" />
                <xs:attribute name="EnvelopeProtocol"
                  type="xs:string" use="optional" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      - <xs:element name="Activity" type="xs:string">
        - <xs:annotation>
            <xs:documentation>Activity name</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="SoapAction" type="xs:string" use="optional" />
    </xs:complexType>
  - <xs:complexType name="GeneralContractType">
    - <xs:sequence>
      - <xs:element name="From">
        - <xs:annotation>
```

```
      <xs:documentation>Sending
        party/service/activity</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
        <xs:element name="FromAddress"
          type="gen:FromAddressType" minOccurs="0" />
      - <xs:element name="SenderDDID" type="xs:string"
          minOccurs="0">
        - <xs:annotation>
            <xs:documentation>DDID of the sender.
              This will not be present if the sender is
              a virtual CP or if the mode is
              client/server</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <xs:element name="To">
  - <xs:annotation>
      <xs:documentation>Receiving
        party/service/activity</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
        <xs:element name="ToAddress"
          type="gen:ToAddressType" />
      - <xs:element name="ReceiverDDID"
          type="xs:string" minOccurs="0">
        - <xs:annotation>
            <xs:documentation>DDID of the receiver.
              DDID of the sender. This will not be
              present if the receiver is a virtual
              CP</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <xs:element name="ErrorHandling">
  - <xs:annotation>
      <xs:documentation>This is a capability parameter in
        activity definition</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
      - <xs:element name="SendAsyncErrorResponseTo"
          type="gen:ServiceActivityType" minOccurs="0">
        - <xs:annotation>
            <xs:documentation>service/activity in
              From party to which the async error
              response should be
              sent</xs:documentation>
```

```
            </xs:annotation>
          </xs:element>
        </xs:sequence>
      - <xs:attribute name="SenderAcceptsAsyncError"
          type="xs:boolean" use="required">
        - <xs:annotation>
            <xs:documentation>Indicates whether the
              sender accepts async error response. This
              only applies to one-way
              messages</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:complexType>
    </xs:element>
  - <xs:element name="DeliveryReceiptHandling">
    - <xs:annotation>
        <xs:documentation>This is a capability parameter in
          activity definition</xs:documentation>
      </xs:annotation>
    - <xs:complexType>
      - <xs:sequence>
        - <xs:element name="SendAsyncDeliveryReceiptTo"
            type="gen:ServiceActivityType" minOccurs="0">
          - <xs:annotation>
              <xs:documentation>service/activity in
                From party to which the delivery
                receipt should be
                sent</xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
      - <xs:attribute name="SenderRequiresDeliveryReceipt"
          type="xs:boolean" use="required">
        - <xs:annotation>
            <xs:documentation>This applies to only one-
              way messages</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="IsSignatureRequiredBySender"
          type="xs:boolean" use="optional" />
        <xs:attribute
          name="IsAsyncDeliveryReceiptAcceptedBySender"
          type="xs:boolean" use="optional" />
      - <xs:attribute
          name="ReceiverCanGenerateAsyncDeliveryReceipt"
          type="xs:boolean" use="optional">
        - <xs:annotation>
            <xs:documentation>Indicates whether the
              receiver can generate a delivery receipt as
              required by the sender. If set to false,
              gateway will generate the delivery receipt
              on behalf of the receiving
              connector</xs:documentation>
          </xs:annotation>
        </xs:attribute>
```

```
    </xs:complexType>
  </xs:element>
- <xs:element name="RequiredMessagePart"
    type="gen:MessagePartInfo" maxOccurs="unbounded">
  - <xs:annotation>
      <xs:documentation>Contains information collected
        from the registry for all the required message
        parts</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="OptionalMessagePart"
    type="gen:MessagePartInfo" minOccurs="0"
    maxOccurs="unbounded">
  - <xs:annotation>
      <xs:documentation>Contains information collected
        from the registry for all the optional message
        parts</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="SendingConnectorCapabilities"
    type="gen:ConnectorCapabilitiesType" minOccurs="0">
  - <xs:annotation>
      <xs:documentation>Describes the list of attributes and
        their associated values for the send side connector.
        This will not be present for non-collaborative
        request and oneway messages</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="ReceivingConnectorCapabilities"
    type="gen:ConnectorCapabilitiesType" minOccurs="0">
  - <xs:annotation>
      <xs:documentation>Describes the list of attributes and
        their associated values for the receive side
        connector. This will not be present for non-
        collaborative response message</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:sequence>
- <xs:attribute name="ChoreographyID" type="xs:anyURI"
    use="optional">
  - <xs:annotation>
      <xs:documentation>Choreography which the service is
        associated with. This only applies to Collaborative
        interactions</xs:documentation>
    </xs:annotation>
</xs:attribute>
- <xs:attribute name="MessageType" use="required">
  - <xs:annotation>
      <xs:documentation>Indicates if the message is request,
        response or oneway</xs:documentation>
    </xs:annotation>
  - <xs:simpleType>
    - <xs:restriction base="xs:string">
        <xs:enumeration value="REQUEST" />
        <xs:enumeration value="RESPONSE" />
```

```
            <xs:enumeration value="ONEWAY" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    - <xs:attribute name="CollaborativeInteraction" type="xs:boolean"
        use="required">
      - <xs:annotation>
          <xs:documentation>Indicates whether it is a collaborative
            or non-collaborative messaging
            paradigm</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    - <xs:attribute name="ICDTimeToLive" type="xs:long" use="required">
      - <xs:annotation>
          <xs:documentation>Time duration after which the cached
            version of this ICD expires. This value is set in the
            config file</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    - <xs:attribute name="MessageTimeToLive" type="xs:long"
        use="required">
      - <xs:annotation>
          <xs:documentation>Time duration after which the message
            will be dropped. This value is set in the activity
            definition</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    - <xs:attribute name="MessageArchived" type="xs:boolean"
        use="required">
      - <xs:annotation>
          <xs:documentation>Indicates whether the message should
            be archived. This is a capability parameter in activity
            definition</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    - <xs:attribute name="BusinessIntelligence" type="xs:boolean"
        use="required">
      - <xs:annotation>
          <xs:documentation>Indicates whether the message is
            available for Business Intelligence purposes. This is a
            capability parameter in activity
            definition</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    - <xs:attribute name="ContractID" type="xs:string" use="required">
      - <xs:annotation>
          <xs:documentation>This contract's ID</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    - <xs:attribute name="QualityOfService" use="required">
      - <xs:simpleType>
        - <xs:restriction base="xs:string">
            <xs:enumeration value="EXACTLYONCE" />
            <xs:enumeration value="BESTEFFORT" />
```

```
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
- <xs:complexType name="FromAddressType">
  - <xs:sequence>
    - <xs:element name="Party" type="xs:anyURI">
      - <xs:annotation>
          <xs:documentation>URI of the collaboration party. This
            will not be present if the sender is an unregistered
            foreign party</xs:documentation>
        </xs:annotation>
      </xs:element>
    - <xs:element name="ServiceActivity"
        type="gen:ServiceActivityType" minOccurs="0">
      - <xs:annotation>
          <xs:documentation>Sending service and activity. This
            will not be present if the from party is not present.
            Also, it will not be present if the message is the
            request part of a request/response message in a
            non-collaborative messaging
            paradigm</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
- <xs:complexType name="ToAddressType">
  - <xs:sequence>
    - <xs:element name="Party" type="xs:anyURI">
      - <xs:annotation>
          <xs:documentation>URI of the collaboration
            party</xs:documentation>
        </xs:annotation>
      </xs:element>
    - <xs:element name="ServiceActivity"
        type="gen:ServiceActivityType" minOccurs="0">
      - <xs:annotation>
          <xs:documentation>Receiving service and activity. This
            will not be present if the message is the response
            part of a request/response in a non-collaborative
            messaging paradigm</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
- <xs:complexType name="ConnectorCapabilitiesType">
  - <xs:sequence>
    - <xs:element name="Attribute" maxOccurs="unbounded">
      - <xs:annotation>
          <xs:documentation>List of
            attributes</xs:documentation>
        </xs:annotation>
      - <xs:complexType>
        - <xs:sequence>
```

```
        - <xs:element name="Value" type="xs:string"
            minOccurs="0" maxOccurs="unbounded">
          - <xs:annotation>
              <xs:documentation>Values for each
                attribute</xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
- <xs:complexType name="MessagePartInfo">
  - <xs:attribute name="PartName" type="xs:string" use="required">
    - <xs:annotation>
        <xs:documentation>Name of the document part. An
          example would be PurchaseOrder</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  - <xs:attribute name="DocIDRequired" type="xs:boolean"
      use="required">
    - <xs:annotation>
        <xs:documentation>Document ID of the part. This
          information is present in the input ICD
          Request</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  - <xs:attribute name="Location" type="xs:string" use="required">
    - <xs:annotation>
        <xs:documentation>Location of the part in the message.
          Possible values are SOAP body, attachment and
          external</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  - <xs:attribute name="MimeType" type="xs:string" use="optional">
    - <xs:annotation>
        <xs:documentation>Specifies the MIME
          type</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  - <xs:attribute name="Root" type="xs:boolean" use="required">
    - <xs:annotation>
        <xs:documentation>Indicates if this is the root
          part</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  - <xs:attribute name="XML" type="xs:boolean" use="required">
    - <xs:annotation>
        <xs:documentation>Indicates if this part is an XML
          message</xs:documentation>
      </xs:annotation>
    </xs:attribute>
</xs:complexType>
```

```
</xs:schema>
```

RoutingContract.XSD

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
edited with XML Spy v4.4 U (http://www.xmlspy.com) by Todd Klaus (Commerce One)
-->
- <xsd:schema
    targetNamespace="publicid:com.commerceone:schemas/soapextension/contract/routing/v1_0/RoutingContract.xsd"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:route="publicid:com.commerceone:schemas/soapextension/contract/routing/v1_0/RoutingContract.xsd" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  - <!--
  imports
  -->
  - <!--
  elements and types
  -->
  - <xsd:element name="Route" type="route:RouteType">
    - <xsd:annotation>
        <xsd:documentation>Routing element in the
          ICD</xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  - <xsd:complexType name="RouteType">
    - <xsd:annotation>
        <xsd:documentation>Defines the list of nodes to be traversed
          from sender to receiver</xsd:documentation>
      </xsd:annotation>
    - <xsd:sequence>
      - <xsd:element name="RouteNode" type="route:RouteNodeType"
          minOccurs="2" maxOccurs="unbounded">
        - <xsd:annotation>
            <xsd:documentation>Nodes in the route. There must be
              at least two nodes in the route (sender and
              receiver)</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  - <xsd:complexType name="RouteNodeType">
    - <xsd:annotation>
        <xsd:documentation>Defines a node in the
          route</xsd:documentation>
      </xsd:annotation>
    - <xsd:sequence>
      - <xsd:element name="EntryChannel" type="route:ChannelType">
        - <xsd:annotation>
            <xsd:documentation>Transport and envelope protocol
              used to reach this node. Becomes ExitChannel when
              route is reversed.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
```

```
- <xsd:element name="ExitChannel" type="route:ChannelType">
  - <xsd:annotation>
      <xsd:documentation>Transport and envelope protocol
        used to exit this node. Becomes EntryChannel when
        route is reversed.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
</xsd:sequence>
- <xsd:attribute name="connector" type="xsd:string" use="required">
  - <xsd:annotation>
      <xsd:documentation>GTW unique name consisting of
        issuing authority prefix, type (always connector here),
        community name, and local name</xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
- <xsd:attribute name="isNative" type="xsd:boolean" use="required">
  - <xsd:annotation>
      <xsd:documentation>Indicates whether this connector is
        running C1 software (CWSP 6.0+)</xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
- <xsd:attribute name="connectorFunction" use="required">
  - <xsd:annotation>
      <xsd:documentation>Specifies the role this connector plays
        in the route at the specified node.</xsd:documentation>
    </xsd:annotation>
  - <xsd:simpleType>
    - <xsd:restriction base="xsd:string">
        <xsd:enumeration value="service-send" />
        <xsd:enumeration value="service-receive" />
        <xsd:enumeration value="hub" />
        <xsd:enumeration value="envelope-gateway" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
- <xsd:attribute name="preICDComputation" type="xsd:boolean"
    use="required">
  - <xsd:annotation>
      <xsd:documentation>Indicates whether this node should
        have already been traversed by the time the ICD
        request was made (i.e., it is prior to the current
        connector/envelope protocol)</xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>
- <xsd:complexType name="ChannelType">
  - <xsd:annotation>
      <xsd:documentation>Defines the transport information needed
        to reach the associated node</xsd:documentation>
    </xsd:annotation>
  - <xsd:attribute name="envelopeProtocol" type="xsd:string"
      use="required">
    - <xsd:annotation>
```

```
        <xsd:documentation>Envelope protocol and version
          associated with this channel</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  - <xsd:attribute name="transportSupportedMessageType"
      use="required">
    - <xsd:annotation>
        <xsd:documentation>Message type supported by this
          channel. One of oneway, request-reponse, or
          both</xsd:documentation>
      </xsd:annotation>
    - <xsd:simpleType>
      - <xsd:restriction base="xsd:string">
          <xsd:enumeration value="oneway" />
          <xsd:enumeration value="request-response" />
          <xsd:enumeration value="both" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  - <xsd:attribute name="transportPhysicalAddress" type="xsd:string"
      use="required">
    - <xsd:annotation>
        <xsd:documentation>transport-specifc address (URL,
          node:queue name, etc)</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  - <xsd:attribute name="transportProtocol" type="xsd:string"
      use="required">
    - <xsd:annotation>
        <xsd:documentation>Transport type (HTTPS, Sonic,
          etc.)</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  - <xsd:attribute name="transportReliable" type="xsd:boolean"
      use="required">
    - <xsd:annotation>
        <xsd:documentation>Indicates whether this transport is
          reliable.</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  - <xsd:attribute name="transportNative" type="xsd:boolean"
      use="required">
    - <xsd:annotation>
        <xsd:documentation>Indicates whether this is a natively
          supported transport. If false, it is handled by a
          transport gateway.</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
</xsd:schema>
```

TransformationContract.XSD

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
edited with XML Spy v4.4 U (http://www.xmlspy.com) by Helen Yuen (Commerce One)
  -->
- <!--
Generated by XML Authority. Conforms to w3c http://www.w3.org/2001/XMLSchema
  -->
- <schema
    targetNamespace="publicid:com.commerceone:schemas/soapextension/contract/transformation/v1_0/TransformationContract.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tpc="publicid:com.commerceone:schemas/soapextension/contract/transformation/v1_0/TransformationContract.xsd"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified"
    version="1.0">
  - <!--
  import namespaces
    -->
  - <!--
  global elements
    -->
  - <element name="TransformationContract"
      type="tpc:TransformationContractType">
    - <annotation>
        <documentation>Transformation Contract Block of the
          ICD</documentation>
      </annotation>
    </element>
  - <complexType name="DocInfoType">
    - <sequence>
        <element name="DocURI" type="xs:anyURI" />
        <element name="DocName" type="xs:string" />
        <element name="Namespace" type="xs:anyURI" />
        <element name="Version" type="xs:string" />
      </sequence>
    </complexType>
  - <complexType name="TransformationContractType">
    - <sequence>
      - <element name="DocumentToTransform"
          type="tpc:DocumentToTransformType"
          maxOccurs="unbounded">
        - <annotation>
            <documentation>Source Document transformation
              information</documentation>
          </annotation>
        </element>
        <element name="ResponseDoc" type="tpc:ResponseDocType"
          minOccurs="0" />
      </sequence>
    </complexType>
  - <complexType name="TransformationMapType">
```

```
- <sequence>
  - <element name="Connector" type="xs:anyURI">
    - <annotation>
        <documentation>Connector GTW name. Specify the
          location where the transformation will
          occur.</documentation>
      </annotation>
    </element>
    <element name="StartDoc" type="tpc:DocInfoType" />
    <element name="EndDoc" type="tpc:DocInfoType" />
  - <element name="CommunityID" type="xs:string">
    - <annotation>
        <documentation>Community ID of where the
          transformation maps are located.</documentation>
      </annotation>
    </element>
    <element name="TransformationMapURI" type="xs:anyURI" />
  </sequence>
</complexType>
- <complexType name="ResponseDocType">
  - <sequence>
      <element name="DocIdURI" type="xs:anyURI" />
      <element name="ColumnNum" type="xs:int" />
    </sequence>
</complexType>
- <complexType name="DocumentToTransformType">
  - <sequence>
    - <element name="SourceDocID" type="xs:anyURI">
      - <annotation>
          <documentation>Source Document ID</documentation>
        </annotation>
      </element>
    - <element name="PartName" type="xs:string">
      - <annotation>
          <documentation>Source Document
            PartID</documentation>
        </annotation>
      </element>
    - <element name="Attachment" type="xs:boolean">
      - <annotation>
          <documentation>Receiver attachment preference
            flag</documentation>
        </annotation>
      </element>
    - <element name="AttachmentPartID" type="xs:string"
        minOccurs="0">
      - <annotation>
          <documentation>Attachment Part ID</documentation>
        </annotation>
      </element>
    - <element name="TransformationMap"
        type="tpc:TransformationMapType" minOccurs="0"
        maxOccurs="unbounded">
      - <annotation>
```

```
            <documentation>Transformation
              instructions</documentation>
          </annotation>
        </element>
      </sequence>
    </complexType>
</schema>
```

SecurityContractKeyInfo.XSD

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
edited with XML Spy v4.4 U (http://www.xmlspy.com) by Symon Chang (Commerce One)
  -->
- <xs:schema
    targetNamespace="publicid:com.commerceone:schemas/soapextension/contract/security/v1_0/SecurityContract.xsd"
    xmlns:sicd="publicid:com.commerceone:schemas/soapextension/contract/security/v1_0/SecurityContract.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified"
    version="1.0">
  - <xs:simpleType name="CollaborationPartyID">
    - <xs:annotation>
        <xs:documentation>This is the Collaboration Partner's
          ID</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string" />
    </xs:simpleType>
  - <xs:simpleType name="KeyUsageTypes">
    - <xs:annotation>
        <xs:documentation>Key is used for signature, encryption,
          and/or authentication.</xs:documentation>
      </xs:annotation>
    - <xs:restriction base="xs:NMTOKENS">
        <xs:enumeration value="AUTHENTICATION" />
        <xs:enumeration value="ENCRYPTION" />
        <xs:enumeration value="SIGNATURE" />
        <xs:enumeration value="SSL" />
      </xs:restriction>
    </xs:simpleType>
  - <xs:simpleType name="KeyAlgorithmTypes">
    - <xs:annotation>
        <xs:documentation>Key is RSA or DSA type of
          key.</xs:documentation>
      </xs:annotation>
    - <xs:restriction base="xs:NMTOKENS">
        <xs:enumeration value="RSA" />
        <xs:enumeration value="DSA" />
      </xs:restriction>
    </xs:simpleType>
  - <xs:simpleType name="AuthenticateModeTypes">
    - <xs:annotation>
        <xs:documentation>The location of where the authentication
          takes place. NONE means neither source nor target
          connector will perform the authentication. This may be the
          case of letting foreign connector to perform the
          authentication.</xs:documentation>
      </xs:annotation>
    - <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="SOURCE" />
        <xs:enumeration value="TARGET" />
```

```
        <xs:enumeration value="NONE" />
      </xs:restriction>
    </xs:simpleType>
  - <xs:element name="PublicKey" type="sicd:PublicKeyType">
    - <xs:annotation>
        <xs:documentation>The Public Key record. Each public key will
          have partyID, KeyInfo, description and
          usages.</xs:documentation>
      </xs:annotation>
    </xs:element>
  - <xs:element name="EncryptionKeyInfo">
    - <xs:annotation>
        <xs:documentation>The KeyInfo that has both PublicKeyID and
          X509Data for encryption.</xs:documentation>
      </xs:annotation>
    - <xs:complexType>
      - <xs:complexContent>
        - <xs:extension base="sicd:KeyInfoType">
            <xs:attribute name="KeyOwner"
              type="sicd:CollaberationPartyID" use="optional" />
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
  - <xs:complexType name="PublicKeyType">
    - <xs:annotation>
        <xs:documentation>The Public Key record, including PartyID,
          KeyInfo, Usages and Description.</xs:documentation>
      </xs:annotation>
    - <xs:sequence>
        <xs:element ref="sicd:PartyID" />
      - <xs:element ref="sicd:EncryptionKeyInfo">
        - <xs:annotation>
            <xs:documentation>The KeyInfo block that has KeyID
              and X509 Data.</xs:documentation>
          </xs:annotation>
        </xs:element>
      - <xs:element ref="sicd:KeyTypeUsage" maxOccurs="4">
        - <xs:annotation>
            <xs:documentation>Key is used for signature,
              encryption, and/or
              authentication.</xs:documentation>
          </xs:annotation>
        </xs:element>
      - <xs:element name="KeyAlgorithm"
          type="sicd:KeyAlgorithmTypes" minOccurs="0">
        - <xs:annotation>
            <xs:documentation>The Key is RSA or DSA
              key</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="sicd:Description" minOccurs="0" />
      - <xs:element name="Location" type="xs:string" minOccurs="0">
        - <xs:annotation>
```

```
            <xs:documentation>The connector ID that key the
              Private Key.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  - <xs:element name="PartyID" type="sicd:CollaberationPartyID">
    - <xs:annotation>
        <xs:documentation>Trading partner ID or Collabration Partner
          ID in UUID format.</xs:documentation>
      </xs:annotation>
    </xs:element>
  - <xs:element name="Description" type="xs:string">
    - <xs:annotation>
        <xs:documentation>The description of the
          key</xs:documentation>
      </xs:annotation>
    </xs:element>
  - <xs:element name="KeyTypeUsage" type="sicd:KeyUsageTypes">
    - <xs:annotation>
        <xs:documentation>Key is used for signature, encryption,
          and/or authentication.</xs:documentation>
      </xs:annotation>
    </xs:element>
  - <xs:element name="KeyInfo">
    - <xs:annotation>
        <xs:documentation>The KeyInfo object is from the XMLDsig
          ds:KeyInfo object. However, within SICD we only use Public
          Key ID field.</xs:documentation>
      </xs:annotation>
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="sicd:PublicKeyID" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  - <xs:element name="PublicKeyID" type="xs:string">
    - <xs:annotation>
        <xs:documentation>The Public Key ID is a unique key ID (UUID
          or from XMKS server).</xs:documentation>
      </xs:annotation>
    </xs:element>
  - <xs:element name="PublicKeyName" type="sicd:PublicKeyNameType">
    - <xs:annotation>
        <xs:documentation>The Name of the Public Key. It is same as
          the PublicKeyID but has owner name as the optional
          attribute.</xs:documentation>
      </xs:annotation>
    </xs:element>
  - <xs:complexType name="PublicKeyNameType">
    - <xs:simpleContent>
      - <xs:extension base="xs:string">
          <xs:attribute name="KeyOwner"
            type="sicd:CollaberationPartyID" use="optional" />
```

```
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
- <xs:complexType name="KeyInfoType">
  - <xs:annotation>
      <xs:documentation>This is for Encryption. The KeyInfo object is
        from the XMLDsig ds:KeyInfo object. However, within SICD
        we only use Public Key ID and X509 Certificate two
        fields.</xs:documentation>
    </xs:annotation>
  - <xs:sequence>
      <xs:element ref="sicd:PublicKeyID" />
    - <xs:element name="X509Data" minOccurs="0">
      - <xs:complexType>
        - <xs:sequence>
            <xs:element name="X509Certificate"
              type="xs:base64Binary" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
- <!--
 Policy Types
  -->
- <xs:complexType name="Abstract_PolicyType" abstract="true">
  - <xs:annotation>
      <xs:documentation>This is the abstract policy for all security
        policy related algorithm. The ID is the Template Name for
        the Algorithm.</xs:documentation>
    </xs:annotation>
    <xs:attribute name="PolicyId" type="xs:string" use="optional" />
  </xs:complexType>
- <xs:complexType name="Abstract_CredentialPolicyType"
    abstract="true">
  - <xs:annotation>
      <xs:documentation>This is the abstract policy for authentication
        credential policy algorithm.</xs:documentation>
    </xs:annotation>
  - <xs:complexContent>
    - <xs:extension base="sicd:Abstract_PolicyType">
      - <xs:sequence>
          <xs:element name="CredentialPolicyAlgorithm"
            type="xs:string" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
- <xs:element name="AuthenticateImplementation" type="xs:string">
  - <xs:annotation>
      <xs:documentation>Optional for different implementation, such
        as SAML, SecureID, or Kerberos.</xs:documentation>
    </xs:annotation>
  </xs:element>
```

```
- <xs:element name="AuthenticateMode"
    type="sicd:AuthenticateModeTypes">
  - <xs:annotation>
      <xs:documentation>The location of where the authentication
        takes place. It can be either SOURCE connector or TARGET
        connector. SOURCE means the sender's local connectors will
        perform SAML Single Sign-On type of authentication.
        TARGET means the connector on the receiving end will
        perform the authentication. NONE means neither source nor
        target connector will perform the authentication. This may
        be the case of letting foreign connector to perform the
        authentication.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:complexType name="AuthenticationCredentialPolicyType">
  - <xs:annotation>
      <xs:documentation>This authentication and credential policy
        will work for Basic and X509.</xs:documentation>
    </xs:annotation>
  - <xs:complexContent>
    - <xs:extension base="sicd:Abstract_CredentialPolicyType">
      - <xs:sequence minOccurs="0">
          <xs:element ref="sicd:AuthenticateMode" />
          <xs:element ref="sicd:AuthenticateImplementation"
            minOccurs="0" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
- <xs:complexType name="AnonymousCredentialPolicyType">
  - <xs:annotation>
      <xs:documentation>This is an anonymous credential policy type
        that has no credential.</xs:documentation>
    </xs:annotation>
  - <xs:complexContent>
    - <xs:restriction base="sicd:Abstract_CredentialPolicyType">
      - <xs:sequence>
          <xs:element name="CredentialPolicyAlgorithm"
            type="xs:string" fixed="Anonymous" />
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
- <xs:complexType name="BasicCredentialPolicyType">
  - <xs:annotation>
      <xs:documentation>This is a basic credential policy type that
        uses ID and password as credential.</xs:documentation>
    </xs:annotation>
  - <xs:complexContent>
      <xs:extension base="sicd:AuthenticationCredentialPolicyType"
        />
    </xs:complexContent>
  </xs:complexType>
- <xs:complexType name="X509CredentialPolicyType">
```

```
- <xs:annotation>
    <xs:documentation>This is a X509 credential policy
      type.</xs:documentation>
  </xs:annotation>
- <xs:complexContent>
    <xs:extension base="sicd:AuthenticationCredentialPolicyType"
      />
  </xs:complexContent>
</xs:complexType>
- <xs:complexType name="BASE64_BINARYCredentialPolicyType">
- <xs:annotation>
    <xs:documentation>This is a BASE64_BINARY_CREDENTIAL
      policy type.</xs:documentation>
  </xs:annotation>
- <xs:complexContent>
  - <xs:extension base="sicd:AuthenticationCredentialPolicyType">
    - <xs:sequence>
        <xs:element name="valueType" type="xs:QName" />
        <xs:element name="encodingType" type="xs:QName" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
- <xs:complexType name="Abstract_EncryptionPolicyType"
    abstract="true">
- <xs:annotation>
    <xs:documentation>This is the abstract policy for Encryption
      policy algorithm.</xs:documentation>
  </xs:annotation>
- <xs:complexContent>
  - <xs:extension base="sicd:Abstract_PolicyType">
    - <xs:sequence>
        <xs:element name="EncryptionPolicyAlgorithm"
          type="xs:string" />
        <xs:element name="EncryptionMethod" type="xs:string"
          />
        <xs:element ref="sicd:KeySize" />
        <xs:element ref="sicd:SymmetryKeySize" minOccurs="0"
          />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
- <xs:complexType name="EncryptionPolicyType">
- <xs:annotation>
    <xs:documentation>This encryption policy will work for both
      XMLEnc and PKCS#7.</xs:documentation>
  </xs:annotation>
- <xs:complexContent>
  - <xs:extension base="sicd:Abstract_EncryptionPolicyType">
    - <xs:sequence>
        <xs:element name="KeyEncryptionMethod"
          type="xs:string" minOccurs="0" />
      </xs:sequence>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
- <xs:element name="KeySize">
  - <xs:annotation>
      <xs:documentation>This is the asymmetry encryption or
        symmetry key size, depends which algorithm is used. For an
        asymmetry case, this will be the asymmetry key size, and
        the symmetry key size is defined on the SymmetryKeySize
        field.</xs:documentation>
    </xs:annotation>
  - <xs:simpleType>
    - <xs:restriction base="xs:short">
        <xs:minInclusive value="56" />
        <xs:maxExclusive value="4096" />
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
- <xs:element name="SymmetryKeySize">
  - <xs:annotation>
      <xs:documentation>This is the symmetry encryption key size, if
        the asymmetry algorithm is used.</xs:documentation>
    </xs:annotation>
  - <xs:simpleType>
    - <xs:restriction base="xs:short">
        <xs:minInclusive value="56" />
        <xs:maxExclusive value="4096" />
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
- <xs:complexType name="XMLEncryptionPolicyType">
  - <xs:annotation>
      <xs:documentation>This will work for any encryption policy
        type.</xs:documentation>
    </xs:annotation>
  - <xs:complexContent>
    - <xs:extension base="sicd:Abstract_EncryptionPolicyType">
      - <xs:sequence>
          <xs:element name="KeyEncryptionMethod"
            type="xs:string"
            default="http://www.w3.org/2001/04/xmlenc#rsa
            -1_5" />
          <xs:element name="DecryptionTransform"
            type="xs:string" minOccurs="0" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
- <xs:complexType name="Abstract_SignaturePolicyType" abstract="true">
  - <xs:annotation>
      <xs:documentation>This is the abstract policy for Digital
        Signature policy algorithm.</xs:documentation>
    </xs:annotation>
  - <xs:complexContent>
```

```
      - <xs:extension base="sicd:Abstract_PolicyType">
        - <xs:sequence>
            <xs:element name="SignaturePolicyAlgorithm"
              type="xs:string" />
            <xs:element name="SignatureAlgorithm"
              type="xs:string" />
            <xs:element name="HashFunction" type="xs:string" />
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  - <xs:complexType name="SignaturePolicyType">
    - <xs:annotation>
        <xs:documentation>This will work for any digital signature
          policy type.</xs:documentation>
      </xs:annotation>
    - <xs:complexContent>
        <xs:extension base="sicd:Abstract_SignaturePolicyType" />
      </xs:complexContent>
    </xs:complexType>
  - <xs:complexType name="XMLDsigPolicyType">
    - <xs:annotation>
        <xs:documentation>This is for XMLDsig
          policy.</xs:documentation>
      </xs:annotation>
    - <xs:complexContent>
      - <xs:extension base="sicd:SignaturePolicyType">
        - <xs:sequence>
            <xs:element name="CanonicalizationMethod"
              type="xs:string" minOccurs="0" />
            <xs:element name="Transform" type="xs:string"
              minOccurs="0" />
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  - <!--
  Message Part
    -->
  - <xs:complexType name="PartElementType">
    - <xs:annotation>
        <xs:documentation>Xpath is used to define the element within
          the part of the message.</xs:documentation>
      </xs:annotation>
    - <xs:simpleContent>
      - <xs:extension base="xs:string">
          <xs:attribute name="Type" type="xs:anyURI" use="optional"
            />
          <xs:attribute name="BlockId" type="xs:short"
            use="optional" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  - <xs:complexType name="MessagePartsType">
```

```
- <xs:annotation>
    <xs:documentation>The part within a message. URI is used to
      define the part.</xs:documentation>
  </xs:annotation>
- <xs:sequence>
  - <xs:element name="PartElement" type="sicd:PartElementType"
      minOccurs="0" maxOccurs="unbounded">
    - <xs:annotation>
        <xs:documentation>The element within the part. It is
          only apply to XML type of message
          part.</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="PartName" type="xs:string" use="required" />
  <xs:attribute name="Type" type="xs:anyURI" use="optional" />
  <xs:attribute name="AlgorithmId" type="xs:string" use="optional" />
  <xs:attribute name="BlockId" type="xs:short" use="optional" />
</xs:complexType>
- <xs:element name="MessagePart" type="sicd:MessagePartsType">
  - <xs:annotation>
      <xs:documentation>The part within the message. The
        AlgorithmId is for this part. If the AlgorithmId is not
        defined, then parent's AlgorithmId will be
        used.</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

SecurityContract.XSD

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
edited with XML Spy v4.4 U (http://www.xmlspy.com) by Symon Chang (Commerce One)
-->
  - <!--
  Security Interop Contract Document

  Created by: Symon Chang
  Copyright 2002 Commerce One, Inc.
-->
- <xs:schema
    targetNamespace="publicid:com.commerceone:schemas/soapextension/contract/security/v1_0/SecurityContract.xsd"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:sicd="publicid:com.commerceone:schemas/soapextension/contract/security/v1_0/SecurityContract.xsd"
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    elementFormDefault="qualified" attributeFormDefault="unqualified"
    version="1.0">
  - <!--
  imports
  -->
    - <!--

      <xs:import
      namespace="publicid:com.commerceone:schemas/soapextension/contract/v1_0/InteroperabilityContract.xsd"
      schemaLocation="http://schemas.commerceone.com/schemas/soapextension/contract/v1_0/InteroperabilityContract.xsd"/>

    -->
    <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion"
      schemaLocation="http://www.oasis-open.org/committees/security/docs/cs-sstc-schema-assertion-01.xsd" />
  - <!--
  includes
  -->
    <xs:include schemaLocation="SecurityContractKeyInfo.xsd" />
  - <!--
   Schema for Security Policies
  -->
  - <!--
   top element
  -->
  - <xs:element name="SecurityContractICD"
      type="sicd:SecurityContractType">
    - <xs:annotation>
        <xs:documentation>The Security Interop Contract agreement. It defines Policies and channels for security policies.</xs:documentation>
      </xs:annotation>
```

```
  </xs:element>
- <!--
Schema for Security Policies
  -->
- <!--
Define Crdetential Policies
  -->
- <xs:element name="BasicCredentialPolicy"
    type="sicd:BasicCredentialPolicyType">
  - <xs:annotation>
      <xs:documentation>The credential and authentication algorithm
        policy for ID and Password.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="X509CredentialPolicy"
    type="sicd:X509CredentialPolicyType">
  - <xs:annotation>
      <xs:documentation>The credential and authentication algorithm
        policy for X.509 Certificate.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="AnonymousCredentialPolicy"
    type="sicd:AnonymousCredentialPolicyType">
  - <xs:annotation>
      <xs:documentation>The credential and authentication algorithm
        policy for no credential.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="BASE64_BINARYCredentialPolicy"
    type="sicd:BASE64_BINARYCredentialPolicyType">
  - <xs:annotation>
      <xs:documentation>The credential and authentication algorithm
        policy for BASE64_BINARY_CREDENTIAL</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="AuthenticationPolicies">
  - <xs:annotation>
      <xs:documentation>The abstraction for credential and
        authentication algorithm policy.</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
        <xs:element ref="sicd:BasicCredentialPolicy" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element ref="sicd:X509CredentialPolicy" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element ref="sicd:BASE64_BINARYCredentialPolicy"
          minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="sicd:AnonymousCredentialPolicy"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <!--
Define Encryption Policies
```

```
  -->
- <xs:element name="EncryptionPolicy"
    type="sicd:EncryptionPolicyType">
  - <xs:annotation>
      <xs:documentation>The encryption algorithm and policy, such
        as PCSK#7, or S/MIME.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="XMLEncryptionPolicy"
    type="sicd:XMLEncryptionPolicyType">
  - <xs:annotation>
      <xs:documentation>The encryption algorithm and policy for
        XMLEnc.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="EncryptionPolicies">
  - <xs:annotation>
      <xs:documentation>The group of encryption algorithms and
        policies for XMLEnc, PCSK#7, or S/MIME. The PolicyID will
        be the TemplateID in the Registry. This ID will be used in
        the Channel Section as AlgorithmID to identify which
        encryption policy algorithm will be used.</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
        <xs:element ref="sicd:XMLEncryptionPolicy" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element ref="sicd:EncryptionPolicy" minOccurs="0"
          maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <!--
Digital Signature Policy
  -->
- <xs:element name="XMLDsigPolicy" type="sicd:XMLDsigPolicyType">
  - <xs:annotation>
      <xs:documentation>The signature algorithm and policy for
        XMLDsig.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="SignaturePolicy" type="sicd:SignaturePolicyType">
  - <xs:annotation>
      <xs:documentation>The signature algorithm and policy for
        XMLDsig, PCSK#7 or S/MIME.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="SignaturePolicies">
  - <xs:annotation>
      <xs:documentation>The group of digital signature algorothms
        and policies for XMLDsig, PCKS#7, or S/MIME. The Policy ID
        will be the TemplateID in the Registry. This Policy ID will be
        used in the Channel Section as AlgorithmID to identify
        which sinature policy algorithm will be
        used.</xs:documentation>
```

```
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
        <xs:element ref="sicd:XMLDsigPolicy" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element ref="sicd:SignaturePolicy" minOccurs="0"
          maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <!--
Non-repudiation
  -->
- <xs:element name="NonRepudiationPolicy"
    type="sicd:SignaturePolicyType"
    substitutionGroup="sicd:NonRepudiationPolicies">
  - <xs:annotation>
      <xs:documentation>The non-repudiationl algorithm and policy
        that use daigital signature.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="NonRepudiationPolicies"
    type="sicd:Abstract_PolicyType" abstract="true">
  - <xs:annotation>
      <xs:documentation>The policy and algorithm for non-
        repudiation of origin.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="NonRepudiationReceiptPolicy"
    type="sicd:SignaturePolicyType"
    substitutionGroup="sicd:NonRepudiationReceiptPolicies">
  - <xs:annotation>
      <xs:documentation>The non-repudiationl algorithm and policy
        that use daigital signature.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="NonRepudiationReceiptPolicies"
    type="sicd:Abstract_PolicyType" abstract="true">
  - <xs:annotation>
      <xs:documentation>The policy and algorithm for non-
        repudiation of receipt.</xs:documentation>
    </xs:annotation>
  </xs:element>
- <xs:element name="SecurityPolicies">
  - <xs:annotation>
      <xs:documentation>The security Policies section. It defines all
        policy related security policies.</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
        <xs:element ref="sicd:AuthenticationPolicies" minOccurs="0"
          />
        <xs:element ref="sicd:SignaturePolicies" minOccurs="0" />
        <xs:element ref="sicd:EncryptionPolicies" minOccurs="0" />
```

```
        <xs:element ref="sicd:NonRepudiationPolicies"
          minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="sicd:NonRepudiationReceiptPolicies"
          minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="sicd:EncryptionKeyInfo" minOccurs="0"
          maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <!--
Schema for Channel
  -->
- <xs:complexType name="KeyAlgorithmType">
  - <xs:annotation>
      <xs:documentation>The root for Integraty and Confidential
        blocks. All these two types of block within the Security
        channel have to have PublicKeyID and AlgorithmId, so does
        the signing and encryption policy within the Credentail
        block.</xs:documentation>
    </xs:annotation>
  - <xs:sequence>
      <xs:element ref="sicd:PublicKeyName" />
    </xs:sequence>
    <xs:attribute name="AlgorithmId" type="xs:string" use="optional" />
  </xs:complexType>
- <xs:complexType name="KeyMessagePartsType">
  - <xs:annotation>
      <xs:documentation>The root for parts in a message. It also
        define the KeyInfo and the algorithm policy for all
        parts.</xs:documentation>
    </xs:annotation>
  - <xs:complexContent>
    - <xs:extension base="sicd:KeyAlgorithmType">
      - <xs:sequence minOccurs="0">
          <xs:element ref="sicd:MessagePart" minOccurs="0"
            maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="SequenceID" type="xs:short"
          use="optional" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
- <xs:element name="Credential">
  - <xs:annotation>
      <xs:documentation>The credentail and authentication polocy.
        Note that the CredentailEncryptionAlgorithm is here. This is
        due to authentication will be preformed before the
        decryption at inbound.</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence minOccurs="0">
      - <xs:choice minOccurs="0">
        - <xs:element name="PartyID"
            type="sicd:CollaberationPartyID" minOccurs="0">
```

```
        - <xs:annotation>
            <xs:documentation>The party ID that is used
              for Basic credentail.</xs:documentation>
          </xs:annotation>
        </xs:element>
      - <xs:element ref="sicd:PublicKeyName" minOccurs="0">
        - <xs:annotation>
            <xs:documentation>The key that is used for
              X.509 credntial.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
    - <xs:element name="CredentialEncryptionAlgorithm"
        type="sicd:KeyAlgorithmType" minOccurs="0">
      - <xs:annotation>
          <xs:documentation>The Encryption Algorithm that
            is used to encrypt the credntial. This will only
            be used when the Authentication mode is
            TARGET.</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="AlgorithmId" type="xs:string"
      use="required" />
    <xs:attribute name="SequenceID" type="xs:short"
      use="optional" />
    <xs:attribute name="DelegationFlag" type="xs:boolean"
      use="optional" default="false" />
  </xs:complexType>
</xs:element>
- <xs:element name="Confidential">
  - <xs:annotation>
      <xs:documentation>The encryption security policy. The
        AlgorithmId will be the tmeplateID from the Registry. If the
        AlgorothmId is defined and no message parts, then the
        whole message will be encrypted. In this case, if there are
        Non-XML parts, then the NonXMLAlgorithmID will be
        defined, too.</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:complexContent>
      - <xs:extension base="sicd:KeyMessagePartsType">
          <xs:attribute name="NonXMLAlgorithmId"
            type="xs:string" use="optional" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
- <xs:element name="Integrity">
  - <xs:annotation>
      <xs:documentation>The digital signature security policy. The
        AlgorithmId will be the tmeplateID from the Registry. If the
        AlgorithmID is defined, and no message parts then the
        whole message will be signed.</xs:documentation>
    </xs:annotation>
```

```
- <xs:complexType>
  - <xs:complexContent>
    - <xs:extension base="sicd:KeyMessagePartsType">
      - <xs:sequence minOccurs="0">
        - <xs:element name="HeaderSignatureAlgorithm"
            type="sicd:KeyAlgorithmType" minOccurs="0">
          - <xs:annotation>
              <xs:documentation>The Signature
                Algorithm that is used to sign the
                header credntial.</xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="NonXMLAlgorithmId"
          type="xs:string" use="optional" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
- <xs:element name="NonRepudiation">
  - <xs:annotation>
      <xs:documentation>The non-repudiation of orgin
        policy.</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
        <xs:element name="NROSignPart"
          type="sicd:KeyMessagePartsType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <xs:element name="NonRepudiationReceipt">
  - <xs:annotation>
      <xs:documentation>The non-repudiation of receipt
        policy.</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:sequence>
        <xs:element name="NRRSignPart"
          type="sicd:KeyMessagePartsType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
- <xs:element name="Authorization">
  - <xs:annotation>
      <xs:documentation>The SAML attribute assertion for the
        sending CP that will be pass to the reciving service. This will
        be shown in the end-to-end security
        channel.</xs:documentation>
    </xs:annotation>
  - <xs:complexType>
    - <xs:simpleContent>
      - <xs:extension base="xs:string">
```

```
            <xs:attribute name="RequireSubscription"
              type="xs:boolean" use="optional" default="false" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    - <!--
      saml:AttributeStatementType">
      -->
    </xs:element>
  - <xs:element name="SecurityContainer">
    - <xs:annotation>
        <xs:documentation>This will be the container for those piggy
          back security related objects.</xs:documentation>
      </xs:annotation>
    - <xs:complexType>
      - <xs:sequence>
          <xs:element name="PiggbackObject" type="xs:anyType"
            minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  - <xs:element name="SecurityChannel">
    - <xs:annotation>
        <xs:documentation>The Security Channel defines the from
          connector and to connector, and what to do within the
          channel, such as authentication, encryption and digital
          signature.</xs:documentation>
      </xs:annotation>
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="sicd:Credential" minOccurs="0" />
          <xs:element ref="sicd:Confidential" minOccurs="0" />
          <xs:element ref="sicd:Integrity" minOccurs="0" />
        - <xs:element ref="sicd:Authorization" minOccurs="0">
          - <xs:annotation>
              <xs:documentation>The SAML attribute assertion
                for the sending CP that will be pass to the
                reciving service. This will be shown in the end-
                to-end security channel.</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element ref="sicd:NonRepudiation" minOccurs="0" />
          <xs:element ref="sicd:NonRepudiationReceipt"
            minOccurs="0" />
        - <xs:element ref="sicd:SecurityContainer" minOccurs="0">
          - <xs:annotation>
              <xs:documentation>This will be the container for
                those piggy back security related
                objects.</xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="channelId" type="xs:string" use="optional"
          />
```

```
      <xs:attribute name="sourceConnector" type="xs:string"
        use="required" />
      <xs:attribute name="targetConnector" type="xs:string"
        use="required" />
    </xs:complexType>
  </xs:element>
- <xs:complexType name="SecurityContractType">
  - <xs:sequence>
      <xs:element ref="sicd:SecurityPolicies" />
      <xs:element ref="sicd:SecurityChannel"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

InteroperabilityContract.XML

```
<?xml version="1.0" ?>
- <!--
edited with XML Spy v4.3 U (http://www.xmlspy.com) by Ernest Beffel (same)
-->
- <InteroperabilityContract
    xmlns="publicid:com.commerceone:schemas/soapextension/contract/v1_0/InteroperabilityContract.xsd"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:general="publicid:com.commerceone:schemas/soapextension/contract/general/v1_0/GeneralContract.xsd"
    xmlns:route="publicid:com.commerceone:schemas/soapextension/contract/routing/v1_0/RoutingContract.xsd"
    xmlns:security="publicid:com.commerceone:schemas/soapextension/contract/security/v1_0/SecurityContract.xsd"
    xmlns:xform="publicid:com.commerceone:schemas/soapextension/contract/transformation/v1_0/TransformationContract.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="publicid:com.commerceone:schemas/soapextension/contract/v1_0/InteroperabilityContract.xsd
    http://schemas.commerceone.com/schemas/soapextension/contract/v1_0/InteroperabilityContract.xsd">
  - <GeneralContract ChoreographyID="ccns:orderManagement"
      MessageType="ONEWAY" CollaborativeInteraction="true"
      ICDTimeToLive="123456" MessageTimeToLive="2147483647"
      MessageArchived="true" BusinessIntelligence="true" ContractID="x-ccns:commerceone.com:CollaborationParty::buyPartyx-ccns:commerceone.com:CollaborationParty::sellParty"
      QualityOfService="EXACTLYONCE">
    - <general:From>
      - <general:FromAddress>
          <general:Party>x-ccns:commerceone.com:CollaborationParty::buyParty</general:Party>
        - <general:ServiceActivity>
            <general:Service Version="1.0" EnvelopeProtocol="C1 SOAP 1.0">A:consumerOrderManagement</general:Service>
            <general:Activity>sendOrder</general:Activity>
          </general:ServiceActivity>
        </general:FromAddress>
        <general:SenderDDID>9f76db48-784d-1000-b0d5-0a0a02030002</general:SenderDDID>
      </general:From>
    - <general:To>
      - <general:ToAddress>
          <general:Party>x-ccns:commerceone.com:CollaborationParty::sellParty</general:Party>
        - <general:ServiceActivity>
            <general:Service Version="1.0" EnvelopeProtocol="C1 SOAP
```

```
            1.0">A:providerOrderManagement</general:Service
            >
          <general:Activity>processOrder</general:Activity>
        </general:ServiceActivity>
      </general:ToAddress>
      <general:ReceiverDDID>9f76db48-784d-1000-b0d5-
        0a0a02030001</general:ReceiverDDID>
    </general:To>
  - <general:ErrorHandling SenderAcceptsAsyncError="true">
    - <general:SendAsyncErrorResponseTo>
        <general:Service Version="1.0" EnvelopeProtocol="C1 SOAP
          1.0">A:consumerOrderManagement</general:Service>
        <general:Activity>sendOrder</general:Activity>
      </general:SendAsyncErrorResponseTo>
    </general:ErrorHandling>
  - <general:DeliveryReceiptHandling SenderRequiresDeliveryReceipt="true"
      IsAsyncDeliveryReceiptAcceptedBySender="true"
      ReceiverCanGenerateAsyncDeliveryReceipt="true">
    - <general:SendAsyncDeliveryReceiptTo>
        <general:Service Version="1.0" EnvelopeProtocol="C1 SOAP
          1.0 A:consumerOrderManagement</general:Service>
        <general:Activity>DeliveryReceiptConsumer</general:Acti
        vity>
      </general:SendAsyncDeliveryReceiptTo>
    </general:DeliveryReceiptHandling>
    <general:RequiredMessagePart PartName="Order" DocIDRequired="true"
      Location="attachment" MimeType="text/xml" Root="true"
      XML="false" />
    <general:RequiredMessagePart PartName="Image"
      DocIDRequired="false" Location="attachment"
      MimeType="image/jpeg" Root="false" XML="false" />
    <general:OptionalMessagePart PartName="someXMLPart"
      DocIDRequired="false" Location="soapbody" MimeType="text/xml"
      Root="false" XML="false" />
  - <general:SendingConnectorCapabilities>
    - <general:Attribute
        Name="Messaging.SupportDeliveryReceiptRequest">
        <general:Value>None</general:Value>
      </general:Attribute>
    - <general:Attribute Name="Messaging.ConversationData">
        <general:Value>rrn:org.soapextensions:schemas/highpe
        rformancesoap/conversationdata/v1_0/ConversationD
        ata</general:Value>
      </general:Attribute>
    - <general:Attribute Name="Messaging.AddressInfo">
        <general:Value>rrn:org.soapextensions:schemas/highpe
        rformancesoap/addressinfo/v1_0/AddressInfo</genera
        l:Value>
      </general:Attribute>
    - <general:Attribute Name="Messaging.MessageIdentity">
        <general:Value>rrn:org.soapextensions:schemas/highpe
```

```
      rformancesoap/messageidentity/v1_0/MessageIdentit
      y</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Archiving.Archiving">
      <general:Value>Yes</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.MessageTimeData">

      <general:Value>rrn:org.soapextensions:schemas/highpe
      rformancesoap/messagetimedata/v1_0/MessageTimeD
      ata</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.Privacy">

      <general:Value>http://schemas.xmlsoap.org/ws/2002/
      04/secext</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.Credential">
      <general:Value>None</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.SecurityAssertion">

      <general:Value>http://schemas.xmlsoap.org/ws/2002/
      04/secext</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.Integrity">

      <general:Value>http://schemas.xmlsoap.org/ws/2002/
      04/secext</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.Manifest">

      <general:Value>rrn:org.soapextensions:schemas:highper
      formancesoap/manifest/v1_0/Manifest</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Transformation.Transformation">
      <general:Value>Yes</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.Reliability">
      <general:Value>None</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.ReturnAddress">
      <general:Value>None</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.MessageEnvelope">
      <general:Value>SOAP WA 1.2</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Archiving.Mining">
      <general:Value>No</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Security.Encryption">
      <general:Value>Message Receiver</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Security.Signing">
      <general:Value>Message Sender</general:Value>
```

```
    </general:Attribute>
  - <general:Attribute Name="Messaging.TestMode">
        <general:Value>rrn:org.soapextensions:schemas/highpe
        rformancesoap/testmode/v1_0/TestMode</general:Valu
        e>
    </general:Attribute>
  - <general:Attribute Name="Messaging.Body">
      <general:Value>Optional</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.ContractData">
        <general:Value>rrn:org.soapextensions:schemas/highpe
        rformancesoap/contractdata/v1_0/ContractData</gene
        ral:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.ReturnDocument">
        <general:Value>rrn:org.soapextensions:schemas/highpe
        rformancesoap/returndocument/v1_0/ReturnDocumen
        t</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.GenerateDeliveryReceipt">
      <general:Value>Yes</general:Value>
    </general:Attribute>
  </general:SendingConnectorCapabilities>
- <general:ReceivingConnectorCapabilities>
  - <general:Attribute
      Name="Messaging.SupportDeliveryReceiptRequest">
      <general:Value>None</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.ConversationData">
        <general:Value>rrn:org.soapextensions:schemas/highpe
        rformancesoap/conversationdata/v1_0/ConversationD
        ata</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.AddressInfo">
        <general:Value>rrn:org.soapextensions:schemas/highpe
        rformancesoap/addressinfo/v1_0/AddressInfo</genera
        l:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.MessageIdentity">
        <general:Value>rrn:org.soapextensions:schemas/highpe
        rformancesoap/messageidentity/v1_0/MessageIdentit
        y</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Archiving.Archiving">
      <general:Value>Yes</general:Value>
    </general:Attribute>
  - <general:Attribute Name="Messaging.MessageTimeData">
        <general:Value>rrn:org.soapextensions:schemas/highpe
```

```
rformancesoap/messagetimedata/v1_0/MessageTimeD
ata</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.Privacy">
<general:Value>http://schemas.xmlsoap.org/ws/2002/
04/secext</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.Credential">
<general:Value>None</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.SecurityAssertion">
<general:Value>http://schemas.xmlsoap.org/ws/2002/
04/secext</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.Integrity">
<general:Value>http://schemas.xmlsoap.org/ws/2002/
04/secext</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.Manifest">
<general:Value>rrn:org.soapextensions:schemas:highper
formancesoap/manifest/v1_0/Manifest</general:Value>
</general:Attribute>
- <general:Attribute Name="Transformation.Transformation">
<general:Value>Yes</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.Reliability">
<general:Value>None</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.ReturnAddress">
<general:Value>None</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.MessageEnvelope">
<general:Value>SOAP WA 1.2</general:Value>
</general:Attribute>
- <general:Attribute Name="Archiving.Mining">
<general:Value>No</general:Value>
</general:Attribute>
- <general:Attribute Name="Security.Encryption">
<general:Value>Message Receiver</general:Value>
</general:Attribute>
- <general:Attribute Name="Security.Signing">
<general:Value>Message Sender</general:Value>
</general:Attribute>
- <general:Attribute Name="Messaging.TestMode">
<general:Value>rrn:org.soapextensions:schemas/highpe
rformancesoap/testmode/v1_0/TestMode</general:Valu
e>
</general:Attribute>
- <general:Attribute Name="Messaging.Body">
<general:Value>Optional</general:Value>
```

```
        </general:Attribute>
      - <general:Attribute Name="Messaging.ContractData">
            <general:Value>rrn:org.soapextensions:schemas/highpe
            rformancesoap/contractdata/v1_0/ContractData</gene
            ral:Value>
        </general:Attribute>
      - <general:Attribute Name="Messaging.ReturnDocument">
            <general:Value>rrn:org.soapextensions:schemas/highpe
            rformancesoap/returndocument/v1_0/ReturnDocumen
            t</general:Value>
        </general:Attribute>
      - <general:Attribute Name="Messaging.GenerateDeliveryReceipt">
          <general:Value>Yes</general:Value>
        </general:Attribute>
      </general:ReceivingConnectorCapabilities>
    </GeneralContract>
  - <RoutingContract>
    - <route:RouteNode connector="x-
        ccns:cup.commerceone.com:connector::buy" isNative="true"
        connectorFunction="service-send" preICDComputation="true">
        <route:EntryChannel envelopeProtocol="C1 SOAP 1.0"
          transportSupportedMessageType="both"
          transportPhysicalAddress="https://uranus.cup.commerceone.c
          om:8433/buy/soap" transportProtocol="https, basic
          authentication" transportReliable="true" transportNative="true"
          />
        <route:ExitChannel envelopeProtocol="C1 SOAP 1.0"
          transportSupportedMessageType="both"
          transportPhysicalAddress="https://uranus.cup.commerceone.c
          om:8433/buy/soap" transportProtocol="https, basic
          authentication" transportReliable="true" transportNative="true"
          />
      </route:RouteNode>
    - <route:RouteNode connector="x-
        ccns:cup.commerceone.com:connector::sell" isNative="true"
        connectorFunction="service-receive" preICDComputation="false">
        <route:EntryChannel envelopeProtocol="C1 SOAP 1.0"
          transportSupportedMessageType="both"
          transportPhysicalAddress="https://saturn.cup.commerceone.c
          om:8433/sell/soap" transportProtocol="https, basic
          authentication" transportReliable="true" transportNative="true"
          />
        <route:ExitChannel envelopeProtocol="C1 SOAP 1.0"
          transportSupportedMessageType="both"
          transportPhysicalAddress="https://saturn.cup.commerceone.c
          om:8433/sell/soap" transportProtocol="https, basic
          authentication" transportReliable="true" transportNative="true"
          />
      </route:RouteNode>
    </RoutingContract>
  - <TransformationContract>
    - <xform:DocumentToTransform>
```

```
      <xform:SourceDocID>x-
        ccns:docid::rrn:org.xcbl:schemas/xcbl/v3_5/xcbl35.xsd:ord
        er:3.5</xform:SourceDocID>
      <xform:PartName>Order</xform:PartName>
      <xform:Attachment>false</xform:Attachment>
    - <xform:TransformationMap>
        <xform:Connector>x-
          ccns:cup.commerceone.com:connector::buy</xform:Con
          nector>
      - <xform:StartDoc>
          <xform:DocURI>x-
            ccns:docid::rrn:org.xcbl:schemas/xcbl/v3_5/xcbl3
            5.xsd:order:3.5</xform:DocURI>
          <xform:DocName>Order</xform:DocName>

            <xform:Namespace>rrn:org.xcbl:schemas/xcbl/v3_5
            /xcbl35.xsd</xform:Namespace>
          <xform:Version>3.5</xform:Version>
        </xform:StartDoc>
      - <xform:EndDoc>
          <xform:DocURI>x-
            ccns:docid::rrn:org.xcbl:schemas/xcbl/v4_0/order
            management/v1_0/ordermanagement.xsd:order:4.
            0.1.0</xform:DocURI>
          <xform:DocName>Order</xform:DocName>

            <xform:Namespace>rrn:org.xcbl:schemas/xcbl/v4_0
            /ordermanagement/v1_0/ordermanagement.xsd</
            xform:Namespace>
          <xform:Version>4.0.1.0</xform:Version>
        </xform:EndDoc>

          <xform:CommunityID>commerceone.com</xform:Commun
          ityID>
        <xform:TransformationMapURI>x-
          ccns:transformationMap:OrderxcbI35Toxcbl4010</xform
          :TransformationMapURI>
      </xform:TransformationMap>
    </xform:DocumentToTransform>
  </TransformationContract>
- <SecurityContract>
  - <security:SecurityPolicies>
    - <security:AuthenticationPolicies>
      - <security:BasicCredentialPolicy PolicyId="P-
          AuthenBasicSource">

            <security:CredentialPolicyAlgorithm>Basic</security:Cre
            dentialPolicyAlgorithm>

            <security:AuthenticateMode>SOURCE</security:Authent
            icateMode>
        </security:BasicCredentialPolicy>
      </security:AuthenticationPolicies>
    - <security:SignaturePolicies>
      - <security:XMLDsigPolicy PolicyId="P-XMLSignatureRSA-MD5-
          EXC14N">
```

<security:SignaturePolicyAlgorithm>**http://www.w3.org/2000/09/xmldsig#**</security:SignaturePolicyAlgorithm>

<security:SignatureAlgorithm>**MD5withRSA**</security:SignatureAlgorithm>
<security:HashFunction>**MD5**</security:HashFunction>

<security:CanonicalizationMethod>**http://www.w3.org/2001/10/xml-exc-c14n#**</security:CanonicalizationMethod>

<security:Transform>**http://msdn.microsoft.com/ws/2002/01/Security#RoutingSignatureTransform**</security:Transform>
</security:XMLDsigPolicy>
</security:SignaturePolicies>
- <security:EncryptionPolicies>
- <security:XMLEncryptionPolicy PolicyId="**P-XMLEncryptAES-128-RSA-2048**">

<security:EncryptionPolicyAlgorithm>**http://www.w3.org/2001/04/xmlenc#**</security:EncryptionPolicyAlgorithm>

<security:EncryptionMethod>**http://www.w3.org/2001/04/xmlenc#aes128-cbc**</security:EncryptionMethod>
<security:KeySize>**2048**</security:KeySize>

<security:SymmetryKeySize>**128**</security:SymmetryKeySize>

<security:KeyEncryptionMethod>**http://www.w3.org/2001/04/xmlenc#rsa-1_5**</security:KeyEncryptionMethod>
</security:XMLEncryptionPolicy>
</security:EncryptionPolicies>
- <security:EncryptionKeyInfo KeyOwner="**x-ccns:commerceone.com:CollaborationParty::sellParty**">

<security:PublicKeyID>**DefaultTestCert**</security:PublicKeyID>
- <security:X509Data>

<security:X509Certificate>**LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tTUlJREZEQ0NBZnlnQXdJQkFnSUVQT0ZQSVRBTkJna3Foa2lHOXcwQkFRVUZBREI2TVFzd0NRWURWUVFHRXdkVlV6RVZNQk1HQTFVRUNoTU1RMjl0YldWeVkyVWdUMjVsTVMwd0t3WURWUVFMRXlSVWFHbHpJRU5CSUdsekxlSWdSWlhOMGFXNW5JSEIxY25CdmMyVnpJRzl1YkhreEpUQWpCZ05WQkFNVUhFTnZiVzFsY21OaElFOXVaU0JVWlhOMElFTkJJRkp2YjNRZ0I6RXdIaGNOTURJd05URTBNVGMxTXpNM1doY05NRE13TlRFME1UWTFNek0zV2pCb01S**</security:X509Certificate>

Z3dGZ1lEVlFRREV3OUVZWFpwWkNCVVpYTjBJREl3T
URJeEVqQVFCZ05WQkFjVENVTjFjR1Z5ZEds
dWJ6RVVNQklHQTFVRUN4TUxSVzVuYVc1bFpYSnBi
bWN4RlRBVEJnTlZCQW9UREVOdmJXMWxjbU5s
SUU5dVpURUxNQWtHQTFVRUJoTUNWVk13Z1o4d0
RRWUpLb1pJaHZjTkFRRUJCUUFEZ1kwQU1JR0pB
b0dCQU5nc2pTQkxjcFp2QnVDQ2lTTHR3RGFkaFZEM
GNLRXJuQ3M2azg5UEhSUGJSMFdYOHBDUzBy
ZWxIMkcyaDMxNU5vNGkzQVNidHZhYmdHellRVFNi
R2EzcWtNYmVLNDZTSGxtTkJOTUp2YUkvMmZV
QlBxdkkzejlLTVJSTGh3eUhCMEdFNmUvSzdnVGZkSU
o0MUJobTZzSmcwYzJqZ041cWt1d3FZQkV4
eWN1MUFnTUJBQUdqT0RBMk1DY0dBMVVkRVFRZ0
1CNkJIRzE1VkdWemRFVnRZV2xzUUdOdmJXMWpa
WEpqWlc5dVpTNWpiMjB3Q3dZRFZSMFBCQVFEQWd
YZ01BMEdDU3FHU0liM0RRRUJCUVVBQTRJQkFR
Q0UrNEVaUWZYZWpmVnBsbXEzZnFtUjJZSGZhczErc
XA0MUg4UWRmNmRESXBiYkZ2OUxocnorYkc2
c2hWQlptMVpYVXphaHl6N2Q3Z2U3V0MxR2FZVjFH
YldFTXJMUkZkeXM2c1VlQkZNbHZuNkZPRjNq
OHdMY3JuN2FFN3pRMEMwa2U5LzVVNVBHTnlaZWV
aUGNLNTlKM0hPdWpzbXUvaENPVW1OOXZVM2M3
MHVjMmhRaE96aExJQ0VlQ2VTRDFCd2hEMXNkdXZ
mNnVOanAzUGp2eUpCaklTeDVxY2UwS25oQmxp
cDR3ejRNTWxpdEtTdkFXSElqR1Bvb0w0N01ac3I4N
3RLamJHaTgxcWJrQ3hiYlZ1dEloYmkzZDRn
aW1Ockc1RXJ0dUUxNmwvRW9GUkJLU2VRTXd2cFd
GUlliN2YreWtKVGE5ZVRLaWF4R2hOcDR4dnc5
LS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQ==</secur
ity:X509Certificate>
</security:X509Data>
</security:EncryptionKeyInfo>
</security:SecurityPolicies>
- <security:SecurityChannel channelId="CHANNEL1" sourceConnector="x-
ccns:cup.commerceone.com:connector::buy" targetConnector="x-
ccns:cup.commerceone.com:connector::sell">
- <security:Credential AlgorithmId="P-AuthenBasicSource"
SequenceID="4">
<security:PartyID>x-
ccns:commerceone.com:CollaborationParty::buyParty</
security:PartyID>
</security:Credential>
- <security:Confidential AlgorithmId="P-XMLEncryptAES-128-RSA-
2048">
<security:PublicKeyName KeyOwner="x-
ccns:commerceone.com:CollaborationParty::sellParty">
DefaultTestCert</security:PublicKeyName>
<security:MessagePart PartName="Order" isOptional="false" />
<security:MessagePart PartName="Image" isOptional="false"
/>
</security:Confidential>
- <security:Integrity AlgorithmId="P-XMLSignatureRSA-MD5-
EXC14N">
<security:PublicKeyName KeyOwner="x-
ccns:commerceone.com:CollaborationParty::buyParty">
DefaultTestCert</security:PublicKeyName>

```
        <security:MessagePart PartName="Order" isOptional="false" />
      </security:Integrity>
    </security:SecurityChannel>
  </SecurityContract>
</InteroperabilityContract>
```

ComputeSecurityContract.XML

```
<?xml version="1.0" ?>
- <prefix_0:SecurityContractICD
    xmlns:prefix_0="publicid:com.commerceone:schemas/soapextension/con
    tract/security/v1_0/SecurityContract.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <prefix_0:SecurityPolicies>
    - <prefix_0:AuthenticationPolicies>
      - <prefix_0:X509CredentialPolicy PolicyId="P-AuthenX.509Source">

            <prefix_0:CredentialPolicyAlgorithm>X.509v3</prefix_0:Cre
            dentialPolicyAlgorithm>

            <prefix_0:AuthenticateMode>SOURCE</prefix_0:Authenticat
            eMode>
        </prefix_0:X509CredentialPolicy>
      </prefix_0:AuthenticationPolicies>
    - <prefix_0:SignaturePolicies>
      - <prefix_0:XMLDsigPolicy PolicyId="P-XMLSignatureRSA-MD5-
          C14N">

            <prefix_0:SignaturePolicyAlgorithm>http://www.w3.org/2
            000/09/xmldsig#</prefix_0:SignaturePolicyAlgorithm>

            <prefix_0:SignatureAlgorithm>MD5withRSA</prefix_0:Sign
            atureAlgorithm>
          <prefix_0:HashFunction>MD5</prefix_0:HashFunction>

            <prefix_0:CanonicalizationMethod>http://www.w3.org/TR
            /2000/CR-xml-c14n-
            20001026</prefix_0:CanonicalizationMethod>

            <prefix_0:Transform>http://msdn.microsoft.com/ws/20
            02/01/Security#RoutingSignatureTransform</prefix_0:
            Transform>
        </prefix_0:XMLDsigPolicy>
      </prefix_0:SignaturePolicies>
    - <prefix_0:EncryptionPolicies>
      - <prefix_0:XMLEncryptionPolicy PolicyId="P-XMLEncrypt3DES-RSA-
          2048">

            <prefix_0:EncryptionPolicyAlgorithm>http://www.w3.org/
            2001/04/xmlenc#</prefix_0:EncryptionPolicyAlgorithm>

            <prefix_0:EncryptionMethod>http://www.w3.org/2001/0
            4/xmlenc#3des-cbc</prefix_0:EncryptionMethod>
          <prefix_0:KeySize>2048</prefix_0:KeySize>

            <prefix_0:KeyEncryptionMethod>http://www.w3.org/200
            1/04/xmlenc#rsa-1_5</prefix_0:KeyEncryptionMethod>
        </prefix_0:XMLEncryptionPolicy>
      </prefix_0:EncryptionPolicies>
    - <prefix_0:EncryptionKeyInfo KeyOwner="x-
        ccns:commerceone.com:CollaborationParty::sellParty">
        <prefix_0:PublicKeyID>DefaultTestCert</prefix_0:PublicKeyID>
```

<prefix_0:X509Data>

<prefix_0:X509Certificate>**LS0tLS1CRUdJTiBDRVJUSUZJQ**
0FURS0tLS0tTUlJREZEQ0NBZnlnQXdJQkFnSUVQT0ZQSV
RBTkJn
a3Foa2lHOXcwQkFRVUZBREI2TVFzd0NRWURWUVFHRX
dKVlV6RVZNQk1HQTFVRUNoTU1RMjl0YldW
eVkyVWdUMjVsTVMwd0t3WURWUVFMRXlSVWFHbHpJR
U5CSUdsekIHWnZjaUIwWlhOMGFXNW5JSEIx
Y25CdmMyVnpJRzl1YkhreEpUQWpCZ05WQkFNVUhFTnZ
iVzFsY21ObElFOXVaU0JVWlhOMElFTkJJ
Rkp2YjNRZ0l6RXdIaGNOTURJd05URTBNVGMxTXpNM1d
oY05NRE13TlRFME1UWTFNek0zV2pCb01S
Z3dGZ1lEVlFRREV3OUVZWFpwWkNCVVpYTjBJREl3TURJ
eEVqQVFCZ05WQkFjVENVTjFjR1Z5ZEds
dWJ6RVVNQklHQTFVRUN4TUxSVzVuYVc1bFpYSnBibWN
4RlRBVEJnTlZCQW9UREVOdmJXMWxjbU5s
SUU5dVpURUxNQWtHQTFVRUJoTUNWVk13Z1o4d0RRW
UpLb1pJaHZjTkFRRUJCUUFEZ1kwQU1JR0pB
b0dCQU5nc2pTQkxjcFp2QnVDQ2lTTHR3RGFkaFZEMGNL
RXJuQ3M2azg5UEhSUGJSMFdYOHBDUzBy
ZWxIMkcyaDMxNU5vNGkzQVNidHZhYmdHellRVFNiR2Ez
cWtNYmVLNDZTSGxtTkJOTUp2YUkvMmZV
QlBxdkkzejlLTVJSTGh3eUhCMEdFNmUvSzdnVGZkSUo0M
UJobTZzSmcwYzJqZ041cWt1d3FZQkV4
eWN1MUFnTUJBQUdqT0RBMk1DY0dBMVVkRVFRZ01CN
kJIRzE1VkdWemRFVnRZV2xzUUdOdmJXMWpa
WEpqWlc5dVpTNWpiMjB3Q3dZRFZSMFBCQVFEQWdYZ0
1BMEdDU3FHU0liM0RRRUJCUVVBQTRJQkFR
Q0UrNEVaUWZYZWpmVnBsbXEzZnFtUjJZSGZhczErcXA0
MUg4UWRmNmRESXBiYkZ2OUxocnorYkc2
c2hWQlptMVpYVXphaHl6N2Q3Z2U3V0MxR2FZVjFHYldF
TXJMUkZkeXM2c1VlQkZNbHZuNkZPRjNq
OHdMY3JuN2FFN3pRMEMwa2U5LzVVNVBHTnlaZWVaUG
NLNTlKM0hPdWpzbXUvaENPVW1OOXZVM2M3
MHVjMmhRaE96aExJQ0VlQ2VTRDFCd2hEMXNkdXZmNn
VOanAzUGp2eUpCaklTeDVxY2UwS25oQmxp
cDR3ejRNTWxpdEtTdkFXSElqR1Bvb0w0N01ac3I4N3RLa
mJHaTgxcWJrQ3hiYlZ1dEloYmkzZDRn
aW1Ockc1RXJ0dUUxNmwvRW9GUkJLU2VRTXd2cFdGUll
iN2YreWtKVGE5ZVRLaWF4R2hOcDR4dnc5
LS0tLS1FTkQgQ0VSVEIGSUNBVEUtLS0tLQ==</prefix_0:
X509Certificate>

</prefix_0:X509Data>

</prefix_0:EncryptionKeyInfo>

</prefix_0:SecurityPolicies>

<prefix_0:SecurityChannel channelId="**CHANNEL1**" sourceConnector="**x-ccns:cup.commerceone.com:connector::buy**" targetConnector="**x-ccns:cup.commerceone.com:connector::sell**">

<prefix_0:Credential AlgorithmId="**P-AuthenX.509Source**" SequenceID="**4**" DelegationFlag="**false**">

<prefix_0:PublicKeyName>**BuyerPublicKey**</prefix_0:PublicKeyName>

</prefix_0:Credential>

<prefix_0:Integrity AlgorithmId="**P-XMLSignatureRSA-MD5-C14N**">

```
      <prefix_0:PublicKeyName
        KeyOwner="OwnerA">BuyerPublicKey</prefix_0:PublicKeyNam
        e>
      <prefix_0:MessagePart PartName="Order" isOptional="false" />
    </prefix_0:Integrity>
  </prefix_0:SecurityChannel>
- <prefix_0:SecurityChannel channelId="CHANNEL2" sourceConnector="x-
    ccns:cup.commerceone.com:connector::centerSell"
    targetConnector="x-
    ccns:cup.commerceone.com:connector::centerSell">
  - <prefix_0:Confidential AlgorithmId="P-XMLEncrypt3DES-RSA-2048">
      <prefix_0:PublicKeyName KeyOwner="x-
        ccns:commerceone.com:CollaborationParty::sellParty">Defa
        ultTestCert</prefix_0:PublicKeyName>
      <prefix_0:MessagePart PartName="Order" isOptional="false" />
      <prefix_0:MessagePart PartName="Image" isOptional="false" />
    </prefix_0:Confidential>
  </prefix_0:SecurityChannel>
</prefix_0:SecurityContractICD>
```

CLAIMS

1. A machine-readable data structure that specifies interoperability data for a consuming service and a providing service, the services exchanging documents via a network, optionally using intermediate connectors, the data structure including:

a route between the services, specified by names of the services and the intermediate connectors and a route among the named services and connectors;

a choreography version to be used for an exchange of messages;

policies for archiving the messages, for assuring reliable delivery of the messages and for requiring a receipt acknowledgement whereby repudiation of receipt can be reduced.

2. The data structure of claim 1, further including a specification of signing requirements for parts of a particular message exchanged between the services and at least one signing algorithm to use.

3. The data structure of claim 1, further including a specification of encryption requirements for parts of a particular message exchanged between the services and at least one encryption algorithm to use.

4. The data structure of claim 1, further including a specification of one or more authentication procedures to use.

5. The data structure of claim 1, further including:

a specification of one or more transformation logics to apply to documents included in a particular message exchanged between the services; and

a specification of whether untransformed copies of the documents should be included with transformed copies of the documents.

6. A machine-readable data structure that specifies interoperability data for a consuming service and a providing service, the services exchanging messages including documents via a network, optionally using intermediate connectors, the data structure including:

a specification of signing requirements for parts of a particular message exchanged between the services and at least one signing algorithm to use;

a specification of encryption requirements for parts of a particular message exchanged between the services and at least one encryption algorithm to use; and

a specification of one or more authentication procedures to use.

7. The data structure of claim 6, further including a route between the services, specified by names of the services and the intermediate connectors and a route among the named services and connectors.

8. The data structure of claim 6, further including a choreography version to be used for an exchange of messages.

9. The data structure of claim 6, further including policies for archiving the messages, for assuring reliable delivery of the messages and for requiring a receipt acknowledgement whereby repudiation of receipt can be reduced.

10. The data structure of claim 6, further including:

a specification of one or more format or protocol transformation logics to apply to documents included in a particular message exchanged between the services; and

a specification of whether untransformed copies of the documents should be included with transformed copies of the documents.

11. A machine-readable data structure that specifies interoperability data for a consuming service and a providing service, the services exchanging messages including documents via a network, optionally using intermediate connectors, the data structure including:

a specification of one or more format or protocol transformation logics to apply to documents included in a particular message exchanged between the services; and

a specification of whether untransformed copies of the documents should be included with transformed copies of the document.

12. The data structure of claim 11, further including a route between the services, specified by names of the services and the intermediate connectors and a route among the named services and connector.

13. The data structure of claim 11, further including a choreography version to be used for an exchange of messages.

14. The data structure of claim 11, further including policies for archiving the messages, for assuring reliable delivery of the messages and for requiring a receipt acknowledgement whereby repudiation of receipt can be reduced.

15. The data structure of claim 11, further including a specification of signing requirements for parts of a particular message exchanged between the services and at least one signing algorithm to use.

16. The data structure of claim 11, further including a specification of encryption requirements for parts of a particular message exchanged between the services and at least one encryption algorithm to use.

17. The data structure of claim 11, further including a specification of one or more authentication procedures to use.

18. A machine-readable data structure that specifies current interoperability data for a consuming service and a providing service, the services exchanging messages including documents via a network, prepared by the process of:

responsive to a request to initiate an exchange of messages between the services, accessing interoperability data for the services;

intersecting the interoperability data for the services; and

for the intersections of interoperability data that produce more than one mutually acceptable option, applying decision rules to select one option.

19. The data structure of claim 18, wherein the decision rules are subscribed to by the services.

20. The data structure of claim 18, wherein the decision rules are adopted by subscription of the services to a trading community.

21. The data structure of claim 18, wherein the interoperability data includes one or more of:

a route between the services, specified by names of the services and the intermediate connectors and a route among the named services and connectors;

a choreography version to be used for an exchange of messages;

policies for archiving the messages, for assuring reliable delivery of the messages and for requiring a receipt acknowledgement whereby repudiation of receipt can be reduced;

a specification of one or more format or protocol transformation logics to apply to documents included in a particular message exchanged between the services; and

a specification of whether untransformed copies of the documents should be included with transformed copies of the documents.

22. The data structure of claim 19, wherein the interoperability data includes one or more of:

a route between the services, specified by names of the services and the intermediate connectors and a route among the named services and connectors;

a choreography version to be used for an exchange of messages;

policies for archiving the messages, for assuring reliable delivery of the messages and for requiring a receipt acknowledgement whereby repudiation of receipt can be reduced;

a specification of one or more format or protocol transformation logics to apply to documents included in a particular message exchanged between the services; and

a specification of whether untransformed copies of the documents should be included with transformed copies of the documents.

23. The data structure of claim 18, wherein the interoperability data includes:

a route between the services, specified by names of the services and the intermediate connectors and a route among the named services and connectors;

a choreography version to be used for an exchange of messages;

policies for archiving the messages, for assuring reliable delivery of the messages and for requiring a receipt acknowledgement whereby repudiation of receipt can be reduced.

24. The data structure of claim 18, wherein the interoperability data includes:

a specification of signing requirements for parts of a particular message exchanged between the services and at least one signing algorithm to use;

a specification of encryption requirements for parts of a particular message exchanged between the services and at least one encryption algorithm to use; and

specification of one or more authentication procedures to use.

25. The data structure of claim 18, wherein the interoperability data includes:

a specification of one or more format or protocol transformation logics to apply to documents included in a particular message exchanged between the services; and

a specification of whether untransformed copies of the documents should be included with transformed copies of the documents.