

Connect5 Orchestration Layer

[Company] Implementation

Project Design

Updated April 27, 2012

# Table of Contents

[Table of Contents 2](#_Toc357779893)

[Overview 4](#_Toc357779894)

[Route Stages 4](#_Toc357779895)

[Visual representation of routes 5](#_Toc357779896)

[Transaction Monitors 5](#_Toc357779897)

[Customization 6](#_Toc357779898)

[Integration Solution 6](#_Toc357779899)

[Multiple PDF selection 7](#_Toc357779900)

[Connect5 Orchestration Layer Routing 7](#_Toc357779901)

[Transaction Monitoring 8](#_Toc357779902)

[Error Route 8](#_Toc357779903)

[Policy Quote Request Route 9](#_Toc357779904)

[Virus Validation Service Request Route (Call Virus Scan) 11](#_Toc357779905)

[Virus Validation Service Response Route 11](#_Toc357779906)

[Turnstile Service Request Route 12](#_Toc357779907)

[Turnstile Service Response Route 13](#_Toc357779908)

[Upload Writer Request Route 15](#_Toc357779909)

[Upload Writer Response Route 15](#_Toc357779910)

[Policy Quote Response Route 17](#_Toc357779911)

[Authentication of Redirect URL 17](#_Toc357779912)

[Additional Information 18](#_Toc357779913)

[A. Processors 18](#_Toc357779914)

[XSLT Processors 18](#_Toc357779915)

[Custom Processors 18](#_Toc357779916)

[B. Sequence Diagrams 19](#_Toc357779917)

[Agency Vendor System [Company] Sequence Diagram 19](#_Toc357779918)

[Multiple PDF Selector UI Sequence Diagram 20](#_Toc357779919)

[Multi-select file upload: 20](#_Toc357779920)

[File Attachment Persistence Sequence Diagram 22](#_Toc357779921)

[C. Connect5 Orchestration Layer Architectural Diagram 23](#_Toc357779922)

[D. Message Security 23](#_Toc357779923)

[Request Message 23](#_Toc357779924)

[Response Message 24](#_Toc357779925)

[Attachments 24](#_Toc357779926)

[E. AWSP Request and Response Samples 24](#_Toc357779927)

[SOAP Request 24](#_Toc357779928)

[SOAP Response 30](#_Toc357779929)

[SOAP Fault 32](#_Toc357779930)

[F. Attachment Processor 33](#_Toc357779931)

[A sample SWA message 33](#_Toc357779932)

[Corresponding MTOM message 34](#_Toc357779933)

[G. Authentication 35](#_Toc357779934)

[H. Virus Validation 36](#_Toc357779935)

[I. Turnstile Interface Structure 37](#_Toc357779936)

[Request: 37](#_Toc357779937)

[Response: 37](#_Toc357779938)

[J. Upload Writer Interface Structure 41](#_Toc357779939)

[Request 41](#_Toc357779940)

[Response 42](#_Toc357779941)

[K. File Attachment Interface 44](#_Toc357779942)

# Overview

The Connect5 Orchestration Layer is a new offering that Agencyport provides to customers that have purchased multiple Agencyport products. The goal of this offering is to seamlessly integrate each product with the flexibility of supporting workflows which are unique to each customer implementation.

What is the Connect5 Orchestration Layer? Connect5 Orchestration Layer provides the ability to take data from different sources, evaluate it, transform it, and route to different systems for further processing. Through a listener, data is relayed for evaluation. It is then prepared for transformation if necessary into a standardized format. Once standardized, it is routed to the target destination or destinations. There may be different destinations based on the outcome of the data transformed. One of the destinations may require the data to be formatted again for what the target system requires. This destination may be the target system or may be another route in the process. If another route, the data from the target will now be the source for the next route. It may take many routes to move the data from the source system to get to the target system. This will be explained further with the workflow defined for the [Company] implementation.

## Route Stages

Here are the stages in which the data can be passed through the Connect5 Orchestration Layer for a given route. Each route is comprised of seven stages. Stages are configurable and customizable with parameters, settings and options which are stored within xml files. For information regarding XSLT and Custom Processors see Additional Information section A Processors.

* Listener – accepts data from a source based on protocol defined and relays it to be evaluated. The framework contains a variety of built-in Listeners such as HTTP/SOAP, JMS/SOAP, HTTP, etc.
* Source Processor – evaluates and prepares the data for transformation. Java Processors can be configured to perform functions such as Authentication and massaging business messages in preparation to be processed as part of the Transform stage.
* Source Format / Transform– transform data into standardized format. This is also where data is split out or combined. The transformation is performed using XSL language.
* Route – defines where to send the data based on data content.Target
* Format / Transform– if target requires format that is not standardized another transformation takes place. The transformation is performed using XSL language.
* Target Processor – prepares data for transport
* Transport – passes data to target based on the protocol defined by the target system. The framework provides a variety of built-in destination transports which can be configured for a particular URI.

*Note: Configuration of the Target destination is currently stored in XML configuration files.*

## Visual representation of routes

This is a visual representation of each of stage of a route. Note: In some cases, Processor stages require no processing. In such cases, they are defined as Noop. In some cases, Transformation stages require no formatting or transformation. In such cases, they are defined as Relay.



## Transaction Monitors

Transaction Monitors are configurable objects that listen to error conditions within a route and takes appropriate actions. For example one can configure transaction monitors for error conditions and send email or route the message to an error route.

# Customization

As mentioned previously, the Connect5 Orchestration Layer interface has built-in processes to take in data, transform it and transport it to a target system. If the workflow cannot be performed with the out of the box functionality, Listeners, Transports, Processors, Transformation Modules, Routing Modules and Transaction Monitors can be customized by extending specified classes. Below is the list of abstract classes.

* AbstractListener for Listener.
* AbstractTransport for Transport.
* AbstrtactProcessors for Processors
* AbstractTransformationModule for Transformation Module
* AbstractRoutingModule for Routing Module
* AbstractTransactionMonitor for Transaction Monitors

# Integration Solution

[Company] is rolling out a solution that will use Agencyport’s Turnstile service with AgencyPortal. Data will come from different sources that [Company] has identified. The sources are from the Agency Vendor System or from AgencyPortal that [Company] has implemented. For more information on the workflow, see sequence diagram under Additional Information section B. Sequence Diagrams. From AgencyPortal, ACORD PDFs and custom forms representing Specialty Lines will be selected and passed to Turnstile for ACORD XML to be returned. The Agency Vendor System will also pass ACORD PDFs and custom forms to Turnstile for ACORD XML to be returned. In addition to these forms, Agency Vendor System will pass ACORD XML based on the ACORD 825 form which will be merged with the ACORD XML returned from Turnstile. The ACORD XML either passed from Turnstile or merged with the ACORD 825 XML will finally be transformed into [Company]’s definition of the Commercial Package Specialty line of business and passed to AgencyPortal. The XML will be consumed by AgencyPortal for a new work item to be created and for the persistence of the file attachments.

With the introduction of Connect5 Orchestration Layer, data from these different sources can be evaluated and transported to the proper destinations required by the [Company] workflow. Connect5 Orchestration Layer will perform the merging of the ACORD XML from Agency Vendor System and Turnstile to be transported on to AgencyPortal. For more information, see Additional Information in section C. Connect5 Orchestration Layer Architectural Diagram.



## Multiple PDF selection

Agencyport will provide the ability to select multiple PDFs to be passed to Turnstile. This capability will be added as part of Connect5 Orchestration Layer and will be implemented as a standalone web page invoked from AgencyPortal. A sequence diagram and overview regarding Multiple PDF selection can be found under Additional Information in section B. Sequence Diagram.

## Connect5 Orchestration Layer Routing

The following describes the steps to implement the workflow defined by [Company]. The steps described are using Connect5 Orchestration Layer as the orchestration to achieve the goals outlined above.

Policy Quote Request Route - *(Authentication)*

Virus Validation Service Request Route *(Call Virus Scan)*

Virus Validation Service Response Route

Turnstile Service Request Route - *(Zip PDFs, Call Turnstile*)

Turnstile Service Response Route - *(Create Empty Transaction, Merge Structured Data with XML, Transform ACORD XML into [Company] XML)*

Upload Writer Request Route *- (Call UploadWriter)*

Upload Writer Response Route *- (Call Persist PDF)*

Policy Quote Response Route - *(Transform messages into ACORD response)*

The following are the details associated with the steps identified above.

**Note: There are some items below that are listed as TBD. The reason for this is that the items have dependencies on decisions external to the Connect5 Orchestration Layer processes. For example, the format in which to send the Virus Scan Request to [Company] has not yet been defined. The goal is to have as many of these resolved as possible as part of the next phase of documentation.**

## Transaction Monitoring

The first module setup in Connect5 Orchestration Layer is Transaction Monitoring. All routes get a transaction monitor to determine if an error with data coming from the source has occurred.

These are stages specific to Transaction Monitoring

* Transaction Monitor – Type: Built in internal Route Trigger Monitor
* Customization – None
* Error Listener: Error Condition Listener

## Error Route

If the Transaction Monitor identifies an error, this is sent to the Error Route which will return the error back to the calling system. For the Error Route the Source is the Error Condition. The Listener used is one that comes as part of Connect5 Orchestration Layer. There are no special processors or transformations required. The final response will be sent back to the original client: Agency Vendor System or AgencyPortal. The Format/Transformation will be XSLT to generate the SOAP Fault request. The Processor used to return the error is a SOAP wrapper. The Transport Type will be the built-in transport Synchronous Response Transport.



* Source – *Error Condition*
* Listener – Type: Built-in Triggerable Listener, Name: *Error Condition Listener*
* Processors – Noop.
* Format/ Transformation Module– Relay.
* Routing Module – Straight through/Null Routing Module.
* Target – *Agency Vendor System*, send synchronous response to the caller.
* Format/ Transformation Module – XSLT to generate SOAP Fault.
* Processor
  + SOAP Wrap Processor This is a Java module provided by the framework that wraps it into a SOAP envelope
* Transport – Type: Synchronous Response Transport.

## Policy Quote Request Route

If an error condition is not encountered, the next step is to perform authentication. The expected sources of this route are Agency Vendor System and AgencyPortal. The Source data will be an AWSP message with PDFs as attachments. For more information regarding Message Security see Additional Information section D. Message Security. The Listener that will be used is a built-in SOAP Web Service. The expected path within the request will contain PolicyQuote. For more information regarding AWSP SOAP Request, Response and Fault see Additional Information section E. AWSP Request and Response Samples. The Processor will be an Attachment processor which extracts the attachments. For more information regarding Attachment Processor see Additional Information section F. Attachment Processor. There is no formatting/ transformation required for this process. The Routing Module being used will be the XPATH routing module. The target for this route is that *Virus Validation Service Request Route.* From any of these sources, there is no special formatting. The target requires different processors based on the source of the request. The Name Space processor will remove all name spaces from the request. The Validation processor differs between Agency Vendor System and AgencyPortal sources. For AgencyPortal; the Validation Processor has a limited number of steps which are identified below. If the source is from Sagitta, the WS Validation Processor is used which contains all of the steps that AgencyPortal follow with some additional steps such as Check Time Stamp, Check User Name, etc. The full list of steps for the WS Validation Processor is listed below. For more information regarding Agent Authentication see Additional Information section G. Authentication. To transport to the Virus Validation Service Request Route an Internal Transport will be used.



* Source – ***Agency Vendor System, AgencyPortal***
  + ***AWSP Message with PDF as attachments***
* Listener – Type: SOAP Web Service Listener, Name***:*** HTTP Post Listener, Request Path – PolicyQuote. Synchronous response.
* Processors –
  + Attachment Processor -This processor extracts attachments.
* Format/ Transformation Module – Relay
* Routing Module – XPATH Routing Module. Use WSA ‘From’ Reference directive to determine the source of the transaction. soap:Envelope/soap:Header/wsa:From/wsa:ReferenceParameters/xs:ClientApp/xs:Org/text()
  1. From Agency Vendor System
* Target – ***Virus Validation Service Request Route***
* Format/ Transformation Module - Relay
* Processor –
  + WS Validation Processor – requires configuration to collect the following configuration. Throws validation exception.
    - Check Time Stamp Expiration, Check Time Stamped Signature
    - Check User Name Token Signature.
    - Verify Security Token, accept X509 only.
    - Key Store File Path, Key Store File Type, Key Store File Password, Server Private Key Password.
    - Check Soap Body Signature.
    - Attachments are treated as entity outside of SOAP Message and it is not part of the signature verification process.
    - Specify custom authenticator class to authenticate user credentials that are in the security token.
    - Build ACSI security profile.
    - SOAP Unwrap processor – Extracts the contents in the SOAP Body.
  + Name Space Processor, Remove all name space.
* Transport – Type: Internal Transport.
  1. From Internal application (PDF upload) AgencyPortal
* Target – ***Virus Validation Service Request Route***
* Format/ Transformation Module - Relay
* Processor –
  + Validation Processor – requires configuration to collect the following configuration. Throws validation exception.
    - Specify custom authenticator class to assert user credentials that are in the security token. TBD
    - Build ACSI security profile.
    - SOAP Unwrap processor – Extracts the contents in the SOAP Body,
  + Name Space Processor, Remove all name space.
* Transport – Type: Internal Transport.

## Virus Validation Service Request Route (Call Virus Scan)

Once the data goes through the Policy Quote Route, the next route is the Virus Validation Service Request Route. This route will determine if there are any viruses within the documents being passed. The Listener used for this route is the Built-in triggerable Listener. There are no processors or formatting/transformation requirements for this route. The target for this route is the [Company] Virus Validation Service Request. There are no special transformations or processors for the Target. The transport mechanism to send the request to the virus service will be a [Company] Custom HTTP transport. A URL will need to be configured along with the credentials. For more information regarding implementation of the Virus Validation see Additional Information section H. Virus Validation. The [Company] Virus Validation Service Request will return a response of Virus Validation Service Response.



* Source – ***Policy Quote Request Route***
* Listener – Type: Built-in Triggerable Listener, Name: Virus Validation Service Request Listener.
* Processors – Noop
* Format/ Transformation Module – Relay
* Routing Module – Straight through/Null Routing Module.
* Target – External ***[Company] Virus Validation Service.*** Send response to ***Virus Validation Service Response Listener***
* Format/ Transformation Module – Relay.
* Processor – Noop
* Transport – Type: [Company] Custom HTTP Transport.
* Configure URL.
* Credentials

## Virus Validation Service Response Route

Based on the outcome of the virus scan there are two potential targets. The source for this route is the External Chub Virus Validation Service Response. This uses a built-in Triggerable Listener. The name of the Listener will be the Virus Validation Service Response Listener. There is no special processor for this source. There is currently no special formatting or transformations required for this route. If there is a virus it goes to the Error Route back to the source system. The Format/Transformation requires XSLT transformation. There is no special processing for the Error Condition Transport. If there is not a virus it proceeds forward to the Turnstile Service Request Route. There is no special processors or formatting for the No Virus target. The Transport for each target will be an Internal Transport type.



* Source – **External *[Company] Virus Validation Service Response.***
* Listener – Type: Built-in Triggerable Listener, Name: Virus Validation Service Response Listener
* Processors – Noop.
* Format/ Transformation Module – Relay
* Routing Module –
* Has virus
* Target – ***Error Condition.*** Return to Error Route noted above.
* Format/ Transformation Module – XSLT Format Error Response.
* Processor - Noop
* Transport – Type: Internal Transport.
* No virus
  + Target – ***Turnstile Service Request Route.*** Move forward to the Turnstile service route
  + Format/ Transformation Module – Relay.
  + Processors - Noop
  + Transport – Type: Internal Transport.

## Turnstile Service Request Route

Once the virus scan is complete and no viruses have been identified, the request is sent to the Turnstile Service Request Route. The source of this route is the Virus Validation Service Response Route. This route uses the Built-in Triggerable Listener and the name will be Turnstile Service Response Listener. There are no special processors or formatting requirement for this source. The target for this route is the External Turnstile Cloud Service Request. There is no special formatting for this target. The processors for this target will be to zip the PDF files as well as create a multipart Turnstile request. The transport mechanism to the Turnstile Cloud Service will be using HTTP/POST. For more information regarding Turnstile Interface Structure see Additional Information section I. Turnstile Interface Structure. The URL to the service as well as user credentials will need to be configured as part of the transport stage.



* Source – ***Virus Validation Service Response Route***
* Listener – Type: Built-in Triggerable Listener, Name: Turnstile Service Request Listener.
* Processors – Noop
* Format/ Transformation Module – Relay
* Routing Module – Straight through/Null Routing Module.
* Target – External ***Turnstile Cloud Service***, Send response to ***Turnstile Service Response Listener.***
* Format/ Transformation Module - Relay
* Processors
  + ZIP PDF
  + Formulate multipart turnstile request
* Transport – Type: HTTP/POST
  + Configure URL.
  + Credentials.

## Turnstile Service Response Route

Once the request is sent to the Turnstile Cloud service, a response with the ACORD XML will be expected. The source for this route will be the External Cloud Turnstile Service. The Listener for the response will be the Built-in Triggerable Listener. The name of the Listener will be Turnstile Service Response Listener. The processor for this source will be an XSLT processor. This processor normalizes the XML based on Agencyport’s Normalized ACORD XML for both the response from Turnstile and the original XML received, if applicable. There are no formatting requirements currently for the Turnstile response. There are three different processors that will be engaged based on the Turnstile Response. These processors will require transformation. The first processor is if the response returns a good confidence score from turnstile along with only one ACORD message. Using an XSLT processor, the response from Turnstile is merged with the original XML where applicable. The second processor is if Turnstile returns a bad score which will return no ACORD message. The Turnstile response will be dropped. If there is an original XML only this will be moved to the next route. If there is no original XML document one will be created to be passed to the next route. The third processor is if Turnstile returns a good score with multiple ACORD messages. This uses an XSLT processor which merges the multi-message response into the one mono-line message. This is then merged with the original XML if available. The language used to merge the data will be XSL. Each processor will send to the same target [Company] Upload Writer Request Route using the same transport mechanism of Internal Transport.



* Source – **External** ***Turnstile Cloud Service***
* Listener – Type: Built-in Triggerable Listener, Name: Turnstile Service Response Listener.
* Processors – XSLT Processors
  + APNAX Normalization on Turnstile response
  + APNAX Normalization on original xml.
* Format/ Transformation Module – Relay
* Routing Module – XpathRoutingModule
* Good Score & 1 PolicyQuoteInqRq (Line of Business)
  + Target – ***[Company] Upload Writer Request Route***
  + Format/ Transformation Module -
    - Processors – XSLT Processors Merge Turnstile response with the original XML
  + Transport – Type: Internal Transport
* Bad Score
  + Target –  ***[Company] Upload Writer Request Route***
  + Format/ Transformation Module – Relay
    - Processor – Drop Turnstile response Drop turnstile response, manufacture new xml when original XML is not available downstream processing. Need spec for developing new xml.
  + Transport – Type: Internal Transport
* Good Score & Multiple PolicyQuoteInqRq (Line of Business)
  + Target –  ***[Company] Upload Writer Request Route***
  + Format/ Transformation Module – Relay
  + Processors – XSLT Processors
    - Multi-line Turnstile response to Mono-line.
    - Merge the output from the above processor with the original XML.
  + Transport – Type: Internal Transport

## Upload Writer Request Route

Once the different processors have been determined, the source for the next route is the Turnstile Service Response Route. This uses the Built-in Triggerable Listener. The name of the Listener will be [Company] Upload Writer Request Listener. There is no special processing or formatting for this source. The target for this route is the External [Company] AgencyPortal Upload Writer. There is no special formatting/transformation for this target. An XSLT processor will be used to produce Connect5 Step Data XML for UploadWriter. For more information regarding UploadWriter Interface see Additional Information section J. Upload Writer Interface Structure. The transport mechanism for this route is HTTP/POST. A URL and user credentials using ACSI methodology will be configured as part of the Transport stage.



* Source – Turnstile Service Response Route
* Listener – Type: Built-in Triggerable Listener, Name: [Company] Upload Writer Request Listener.
* Processors – Noop
* Format/ Transformation Module – Relay
* Routing Module – Straight through/Null Routing Module.
* Target: ***External [Company] AgencyPortal Upload Writer***, Send response to ***[Company] Upload Writer Response Listener.***
* Format/ Transformation Module – Relay
* Processors – XSLT Processors
  + Produce Connect5 Step Data xml using Connect5.xsd to be processed by UploadWriter
* Transport – Type: HTTP/POST
  + Configure URL.
  + Credentials/ACSI.

## Upload Writer Response Route

Once the XML has been consumed and the request persisted in AgencyPortal, a response will be sent back to Connect5 Orchestration Layer. The source of this response will be External [Company] Upload Writer. The listener for this route is Built-in Triggerable listener. The name of the Listener will be [Company] Upload Writer Response Listener. There are no special processors for this route. There are currently no special formatting requirements for this route. The routing of data will be determined based on the outcome of the response from AgencyPortal. If an error in creating the Work Item in AgencyPortal takes place, the target will be the Error Condition route. There is special processing to formulate and invalid work item response. There is no special processor. If there are no errors, the target will be the External [Company] Persist PDF. For more information regarding the File Attachment Interface implementation in AgencyPortal, see Additional Information section K. File Attachment Interface**.** There is no special format for this target. There will be a processor for Format Archive request. The transport mechanism for this route is HTTP/POST. A URL and user credentials using ACSI methodology will be configured as part of the Transport stage.

File Attachment persistence

Agencyport will provide the ability to persist the file attachments that are associated with the request in AgencyPortal. For more information on File Attachment Persistence, see sequence diagram under Additional Information section C. Sequence Diagrams.



* Source: ***External [Company] Upload Writer***
* Listener – Type: Built in Triggerable Listener, Name: *[Company] Upload Writer Response Listener*
* Processors: None
* Format/ Transformation Module – Relay
* Routing Module – XPATH Routing Module.
  + Invalid Work Item
    - Target – ***Error Condition.***
    - Format/ Transformation Module
      * Formulate invalid work item response to be handled by error condition route
    - Processor - Noop
    - Transport – Type: Internal Transport***.***
  + Valid Work Item
    - Target: ***External [Company] Persist PDF***, Send response to ***Policy Quote Response Listener.***
    - Format/ Transformation Module – Relay
    - Processors
      * Format Archive request
    - Transport – Type: HTTP/POST
      * Configure URL.
      * Credentials/ACSI.

## Policy Quote Response Route

Once the file attachments are persisted in AgencyPortal, the URL to launch AgencyPortal is sent back to the source system. The source for this route is [Company] Persist PDF. The listener for this route is Built-in Triggerable listener. The name of the Listener is the Policy Quote Response Listener. The processors for this route will be to retrieve UploadWriter response from the transaction attribute and set it as the data stream. There are currently no special formatting requirements for this route. The target is either, the Agency Vendor System or AgencyPortal. The formatting for this target will be to format the response based on the caller. The processors to return to these targets are encrypting the User ID and password in the URL as well as a Soap Processor configuration. The formatting for the target will the format response to the caller. The Transport will be a Synchronous Response transport.



* Source: External ***[Company] Persist PDF***
* Listener – Type: Built-in Triggerable Listener, Name: Policy Quote Response Listener.
* Processors – Retrieve UploadWriter response from the transaction attribute and set it as data stream
* Format/ Transformation Module – Relay
* Routing Module – Straight through/Null Routing Module.
* Target – ***Agency Vendor System, AgencyPortal***, send synchronous response to the caller.
* Format/ Transformation Module
  + Format response to the caller
* Processor –
* Encrypt User Id and password in Portal URL
* SOAP Wrap Processor Transport – Type: Synchronous Response Transport.

## Authentication of Redirect URL

A discussion regarding how to handle authentication for the redirect to launch into AgencyPortal is required. Once this discussion takes place the steps will be outlined, if necessary.

# Additional Information

## A. Processors

### XSLT Processors

XSLT processors are built in java processors that can be configured to use a given XSLT for transforming data.Attachment processors

* Be able to handle multiple attachments
* Downstream processors should be able to access the binary attachments through Data Handlers, instead of the base 64 encoded stream in memory.
* The location for the attachment should be with-in the application context, so that we don’t have to configure explicitly.

### Custom Processors

The Connect5 Orchestration Layer supports implementation of custom processors. In order to implement these, the developer must implement the processData method. Optionally they implement getConfigurationDescriptors method which returns ConfigurationDescriptors. Data to be processed is retrieved from the TranactionData by TranactionData.getDataStream method, after processing, the processed data is set to the TransactionData by TranactionData.setDataStream. Processors can be designed and implemented to collect configuration information when authoring a route, in such case the processors getConfigurationDescriptors must return a valid ConfigurationDescriptors. The ‘Processors and TransactionData’ class diagram below can be used as a reference.

## B. Sequence Diagrams

### Agency Vendor System [Company] Sequence Diagram



### Multiple PDF Selector UI Sequence Diagram

### Multi-select file upload:

Scope and guiding principles:

* Provides a multi file selection upload widget that is not tightly coupled with AgencyPortal applications eliminating dependencies on SDK version.
* Provides capability to auto launch the work item after the response is received from Connect5 Orchestration Layer.
* This widget will provide a way to upload PDF/TIFF files to Connect5 Orchestration Layer. Other files types are out of scope. Collection and passing of additional user entered data is out of scope.
* It is assumed that the portal and Connect5 Orchestration Layer will be deployed in the same domain which facilitates exchange of cookies for the purpose of passing authentication information. If the applications are not in the same domain a different mechanism will be devised to exchange credentials.

High Level Approach:

* When the user clicks on the link an iframe is created and a call is made to the UplaodFileServlet which will redirect the request to the Connect5 Orchestration Layer UI.
* The UploadFileServlet will add cookies to the http response and redirect to the Connect5 Orchestration Layer UI.
* The Connect5 Orchestration Layer security filter will authenticate the user against the carrier security service using the cookies received and build the profile. If authentication fails, an error message will be displayed, otherwise a file selection page is rendered.
* User selects one or more files and clicks Submit.
* A soap message is created with the uploaded files and sent to the PolicyQuote listener. This interaction would follow the request response structures defined in section D. The Client app org node will be set to ‘[Company]’ soap:Envelope/soap:Header/wsa:From/wsa:ReferenceParameters/xs:ClientApp/xs:Org/text()
* For a successful upload, a workitem URL will be launched. In the event of a failure, error message(s) will be displayed.

Potential changes to portal:

* + Add an anchor link to portal screen (Screen outside of transaction screens).
  + Include the Servlet (UploadFileServlet) provide by Connect5 Orchestration Layer and add web.xml entries for the same
  + Include javascript file that will launch the UI in an iframe.

This diagram refers to the Multiple PDF selection that will be invoked from AgencyPortal.

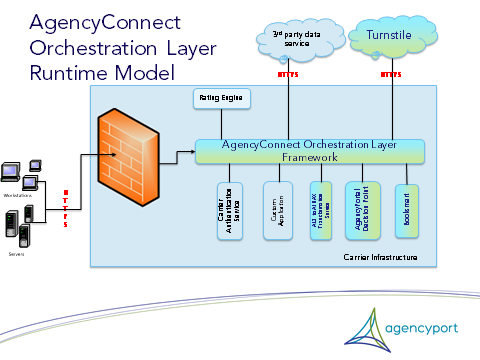


### File Attachment Persistence Sequence Diagram

This diagram outlines what occurs as part of the [Company] Upload Writer Response Route.



## C. Connect5 Orchestration Layer Architectural Diagram



## D. Message Security

### Request Message

Following are some of the key AWSP specifications that will be implemented to address Integrity, Confidentiality, Accountability and Use Control security objectives.

* Senders MUST specify the soap:mustUnderstand attribute on the <wsse:Security> container element – RR7210.
* Message Timestamp has to be included in the signature to protect end-to-end integrity protection and authentication of timestamp – RR0603.
* SOAP Body must be included in the sender’s signature and SOAP body must be signed in totality – RR7601.
* Only X509 certificates must be used to provide cryptographic keys for signatures – RR7501.
* wsse:Nonce & wsu:Created must be included in the Username Token – RR7401.
* wsse:UsernameToken aggregate must be encrypted and signed – RR7403 & RR7404.

### Response Message

* Response message will not be encrypted or signed.

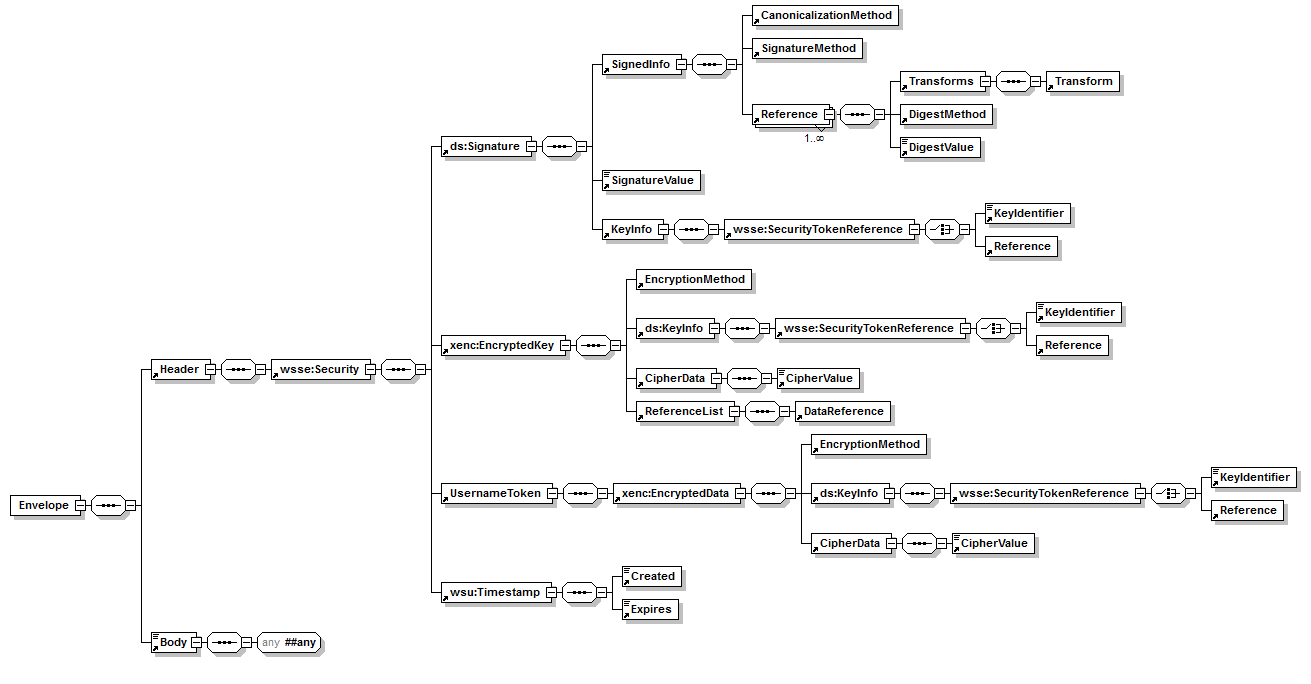
### Attachments

* Signing and Encrypting of attachment is outside the scope of this implementation.

## E. AWSP Request and Response Samples

### SOAP Request

Below is the schema and sample SOAP request with security header decorations. ACORD XXXInqRq will be the root node for the business message.

* **Schema**:  
  
* **Sample**:

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" xmlns:xs="http://www.ACORD.org/standards/PC\_Surety/ACORD1/XMLSchema">

<soap:Header>

<!--wsa:Action routes the message to the target service port; allowed values are defined in the WSDL File in a <wsdl:binding> section.-->

<wsa:Action wsu:Id="MyAction" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://www.agencyport.com/PC\_Surety/ACORD1/PolicyQuote/CommlPkgPolicyQuoteInqRq</wsa:Action>

<!--wsa:MessageID must be equal to the payload's UUID-->

<wsa:MessageID wsu:Id="MyMessageId" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">uuid:7318DA12-384D-43A1-BC98-775F15F26CA3</wsa:MessageID>

<!--wsa:To represents the address of the intended receiver of the message and typically equals one of the port addresses defined in the WSDL file in the <wsdl:service> section; for PCS V1 implementations, it is assumed that SPName information is embedded in this URI.-->

<wsa:To wsu:Id="MyDestination" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://www.agencyport.com/Connect5 Orchestration Layer/http-post/PolicyQuote</wsa:To>

<!--wsa:From represents the endpoint from which the message originated, the ultimate issuer of the message.-->

<wsa:From wsu:Id="MyOrigin" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsa:Address>http://www.source-client.com/Agent/ClientOne</wsa:Address>

<!--If required, one instance of <wsa:ReferenceParameters> can be used to convey specific information from the PCS Framework in their original format; <wsa:ReferenceParameters> may contain a number of individual parameters that are associated with the endpoint to facilitate a particular interaction -->

<wsa:ReferenceParameters>

<xs:SignonRoleCd>Agent</xs:SignonRoleCd>

<xs:PIN>1234</xs:PIN>

<xs:ClientApp>

<xs:Org>Best Vendor</xs:Org>

<xs:Name>Best Software</xs:Name>

<xs:Version>21.c</xs:Version>

</xs:ClientApp>

<xs:GenSessKey>1</xs:GenSessKey>

</wsa:ReferenceParameters>

</wsa:From>

<!-- or, if a synchronous response is required, must be equal to the URI "http://www.w3.org/2005/08/addressing/anonymous" -->

<wsa:ReplyTo wsu:Id="MyReply" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>

</wsa:ReplyTo>

<wsse:Security soap:mustUnderstand="true" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="CertId-27F5C9FBDAD6526E7813323139066013" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">VMeupTFga1V</wsse:BinarySecurityToken>

<ds:Signature Id="Signature-4" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

<ds:SignedInfo>

<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

<ds:Reference URI="#id-5">

<ds:Transforms>

<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

</ds:Transforms>

<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#md5"/>

<ds:DigestValue>4n1OpYm3jVDZpNStpcUDOQ==</ds:DigestValue>

</ds:Reference>

<ds:Reference URI="#UsernameToken-2">

<ds:Transforms>

<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

</ds:Transforms>

<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#md5"/>

<ds:DigestValue>fyoghFDv98SA//B+l9SryQ==</ds:DigestValue>

</ds:Reference>

<ds:Reference URI="#Timestamp-1">

<ds:Transforms>

<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

</ds:Transforms>

<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#md5"/>

<ds:DigestValue>bR0oItKmSzezVJ1VqXWJcw==</ds:DigestValue>

</ds:Reference>

</ds:SignedInfo>

<ds:SignatureValue>eDExTvjQ==</ds:SignatureValue>

<ds:KeyInfo Id="KeyId-27F5C9FBDAD6526E7813323139066164">

<wsse:SecurityTokenReference wsu:Id="STRId-27F5C9FBDAD6526E7813323139066165" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsse:Reference URI="#CertId-27F5C9FBDAD6526E7813323139066013" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>

</wsse:SecurityTokenReference>

</ds:KeyInfo>

</ds:Signature>

<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="27F5C9FBDAD6526E7813323139057671" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">MIICxzCCAa+ =</wsse:BinarySecurityToken>

<xenc:EncryptedKey Id="EncKeyId-27F5C9FBDAD6526E7813323139057962">

<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1\_5"/>

<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

<wsse:SecurityTokenReference>

<wsse:Reference URI="#27F5C9FBDAD6526E7813323139057671" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>

</wsse:SecurityTokenReference>

</ds:KeyInfo>

<xenc:CipherData>

<xenc:CipherValue>AlhAHKNbF7o65P0t7ZlXdGMCu</xenc:CipherValue>

</xenc:CipherData>

<xenc:ReferenceList>

<xenc:DataReference URI="#EncDataId-3"/>

</xenc:ReferenceList>

</xenc:EncryptedKey>

<wsse:UsernameToken wsu:Id="UsernameToken-2" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<xenc:EncryptedData Id="EncDataId-3" Type="http://www.w3.org/2001/04/xmlenc#Content">

<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>

<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

<wsse:SecurityTokenReference>

<wsse:Reference URI="#EncKeyId-27F5C9FBDAD6526E7813323139057962"/>

</wsse:SecurityTokenReference>

</ds:KeyInfo>

<xenc:CipherData>

<xenc:CipherValue>ixhl47aa3cx..</xenc:CipherValue>

</xenc:CipherData>

</xenc:EncryptedData>

</wsse:UsernameToken>

<wsu:Timestamp wsu:Id="Timestamp-1" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsu:Created>2012-03-21T07:11:44Z</wsu:Created>

<wsu:Expires>2012-03-21T07:11:47Z</wsu:Expires>

</wsu:Timestamp>

</wsse:Security>

</soap:Header>

<soap:Body wsu:Id="id-5" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<xs:CommlPkgPolicyQuoteInqRq>

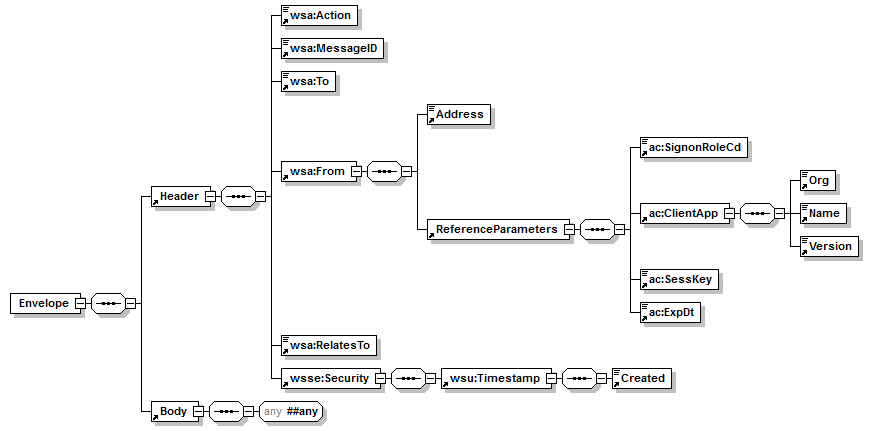
</xs:CommlPkgPolicyQuoteInqRq>

</soap:Body>

</soap:Envelope>

### SOAP Response

Below is the schema and sample SOAP response with security header decorations. ACORD XXXInqRs will be the root node for the response business message.

* **Schema:**
* **Sample:**

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns="http://www.ACORD.org/standards/PC\_Surety/ACORD1/xml/">

<soap:Header>

<!--wsa:Action routes the message into the client system; allowed values are defined in the WSDL File in a <wsdl:binding> section.-->

<wsa:Action wsu:Id="MyAction">http://www.agencyport.com/PC\_Surety/ACORD1/PolicyQuote/CommlPkgPolicyQuoteInqRs</wsa:Action>

<!--wsa:MessageID must be different from the payload's UUID-->

<wsa:MessageID wsu:Id="MyMessageId">uuid:ea9c0cd2-d996-4c23-b21a-c7827eed1fc2</wsa:MessageID>

<!--wsa:To represents the address of the Response EndPoint. It must be equal to the value of <wsa:ReplyTo> in the request message.-->

<wsa:To wsu:Id="MyDestination">http://www.source-client.com/Agent/ClientOne</wsa:To>

<!--wsa:From represents the endpoint from which the message originated. It must be equal to the value of <wsa:To> in the request message.-->

<wsa:From wsu:Id="MyOrigin">

<wsa:Address>http://www.agencyport.com/Connect5 Orchestration Layer/http-post/PolicyQuote</wsa:Address>

<!--If required, one instance of <wsa:ReferenceParameters> can be used to convey specific information from the PCS Framework in their original format; <wsa:ReferenceParameters> may contain a number of individual parameters that are associated with the endpoint to facilitate a particular interaction -->

<wsa:ReferenceParameters>

<ac:SignonRoleCd>Agent</ac:SignonRoleCd>

<ac:ClientApp>

<ac:Org>Best Vendor</ac:Org>

<ac:Name>Best Software</ac:Name>

<ac:Version>21.c</ac:Version>

</ac:ClientApp>

<ac:SessKey>7318DA12-384D-43A1-BC98-775F15F26CA9</ac:SessKey>

<ac:ExpDt>2008-04-02T02:24:32Z</ac:ExpDt>

</wsa:ReferenceParameters>

</wsa:From>

<!--wsa:RelatesTo conveys the Id of the request message; it must be equal to the value of <wsa:MessageID> in the request message.-->

<wsa:RelatesTo>uuid:7318DA12-384D-43A1-BC98-775F15F26CA3</wsa:RelatesTo>

<wsse:Security>

<!--wsu:Timestamp specifies Date and Time of the security semantics.-->

<wsu:Timestamp>

<wsu:Created>2003-07-16T01:24:32Z</wsu:Created>

</wsu:Timestamp>

<!--ds:Signature is optional (sender's message signature) <ds:Signature/> -->

<!--NOTE: wsse:UsernameToken must not be specified in a response message.-->

</wsse:Security>

</soap:Header>

<soap:Body wsu:Id="id-5" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

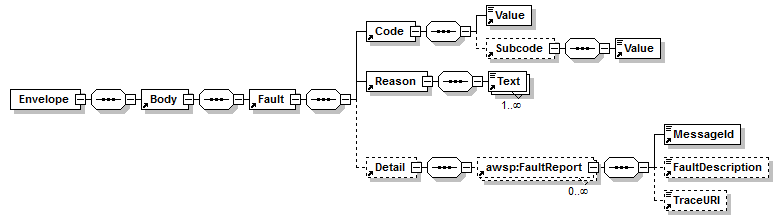
<CommlPkgPolicyQuoteInqRs xmlns="<http://www.ACORD.org/standards/PC_Surety/ACORD1/XMLSchema>" ></CommlPkgPolicyQuoteInqRs>

</soap:Body>

</soap:Envelope>

### SOAP Fault

Refer to AWSP section 7.5 for fault reporting structure. See schema and sample XML below.

* **Schema:**
* **Sample:**

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">

<soap:Body>

<soap:Fault>

<soap:Code>

<soap:Value>soap:Sender</soap:Value>

<soap:Subcode>

<soap:Value xmlns:awsp="http://www.acord.org/schema/data/awsp/2">awsp:InvalidTransactionInformation</soap:Value>

</soap:Subcode>

</soap:Code>

<soap:Reason>

<soap:Text xml:lang="en">Invalid Transaction Information</soap:Text>

</soap:Reason>

<soap:Detail>

<awsp:FaultReport xmlns:awsp="http://www.acord.org/schema/data/awsp/2">

<awsp:MessageId>7318DA12-384D-43A1-BC98-775F15F26CA3</awsp:MessageId>

<awsp:FaultDescription>The element 'TXLifeRequest' in namespace 'http://ACORD.org/Standards/Life/2' has invalid child element 'TransExeDate' in namespace

'http://ACORD.org/Standards/Life/2'. List of possible elements expected: 'TransType' in namespace 'http://ACORD.org/Standards/Life/2'.</awsp:FaultDescription>

<awsp:TraceURI>https://acme.com/WebServices/Trace/7318DA12-384D-43A1-BC98-775F15F26CA3.htm</awsp:TraceURI>

</awsp:FaultReport>

</soap:Detail>

</soap:Fault>

</soap:Body>

</soap:Envelope>

## F. Attachment Processor

### A sample SWA message

POST /PolicyQuote HTTP/1.1

Content-Type: multipart/related; type="application/soap+xml"; action="[http://www.agencyport.com/PC\_Surety/ACORD1/PolicyQuote/CommlPkgPolicyQuoteInqRq](http://www.agencyport.com/PC_Surety/ACORD1/PolicyQuote/HomePolicyQuoteInqRq)"; boundary="uuid:698d820a-a174-47fe-9fe8-53fcabeb81b7"; start="<[root.message@cxf.apache.org](mailto:root.message@cxf.apache.org)>"; start-info="application/soap+xml"; action="[http://www.agencyport.com/PC\_Surety/ACORD1/PolicyQuote/CommlPkgPolicyQuoteInqRq](http://www.agencyport.com/PC_Surety/ACORD1/PolicyQuote/HomePolicyQuoteInqRq)"

Accept: \*/\*

User-Agent: Apache CXF 2.5.2

Cache-Control: no-cache

Pragma: no-cache

Host: 127.0.0.1:9001

Connection: keep-alive

Transfer-Encoding: chunked

ff9

--uuid:698d820a-a174-47fe-9fe8-53fcabeb81b7

Content-Type: text/xml; charset=UTF-8; type="application/soap+xml"; action="[http://www.agencyport.com/PC\_Surety/ACORD1/PolicyQuote/CommlPkgPolicyQuoteInqRq](http://www.agencyport.com/PC_Surety/ACORD1/PolicyQuote/HomePolicyQuoteInqRq)";

Content-Transfer-Encoding: binary

Content-ID: <[root.message@cxf.apache.org](mailto:root.message@cxf.apache.org)>

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope</CommlPkgPolicyQuoteInqRq></soap:Body></soap:Envelope>

--uuid:698d820a-a174-47fe-9fe8-53fcabeb81b7

Content-Type: application/pdf

### Corresponding MTOM message

POST /PolicyQuote HTTP/1.1

Content-Type: multipart/related; type="application/xop+xml"; boundary="uuid:e213b93a-2e70-49f7-b334-48ed0c2d45fd"; start="<[root.message@cxf.apache.org](mailto:root.message@cxf.apache.org)>"; start-info="application/soap+xml"; action="[http://www.agencyport.com/PC\_Surety/ACORD1/PolicyQuote/CommlPkgPolicyQuoteInqRq](http://www.agencyport.com/PC_Surety/ACORD1/PolicyQuote/HomePolicyQuoteInqRq)"

Accept: \*/\*

User-Agent: Apache CXF 2.5.2

Cache-Control: no-cache

Pragma: no-cache

Host: 127.0.0.1:9001

Connection: keep-alive

Transfer-Encoding: chunked

ff9

--uuid:e213b93a-2e70-49f7-b334-48ed0c2d45fd

Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"; action="[http://www.agencyport.com/PC\_Surety/ACORD1/PolicyQuote/CommlPkgPolicyQuoteInqRq](http://www.agencyport.com/PC_Surety/ACORD1/PolicyQuote/HomePolicyQuoteInqRq)";

Content-Transfer-Encoding: binary

Content-ID: <[root.message@cxf.apache.org](mailto:root.message@cxf.apache.org)>

<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>"><soap:Body><CommlPkgPolicyQuoteInqRq xmlns="<http://www.ACORD.org/standards/PC_Surety/ACORD1/XMLSchema>"/></soap:Body></soap:Envelope>

--uuid:e213b93a-2e70-49f7-b334-48ed0c2d45fd

Content-Type: application/pdf

## G. Authentication

Implement com.agencyport.substrate.security.IAuthenticator#performAuthentication method. Set the AuthenticatorClass configuration parameter in the WS Validation Processor.

The parameters for performAuthentication method are

* transactionData is the container that has all the pertinent information related to transaction in context.
* userId that's in the WS-Security Username token.
* password that's in the WS-Security Username token.

The method throws com.agencyport.substrate.exception.UserAuthenticationFailedException when the user cannot be authenticated irrespective of underlying reason.

On successful authentication, set the user's security token in the TransactionData attribute - 'com.agencyport.securitytoken'.

## H. Virus Validation

Implement com.agencyport.substrate.module.ITransport#execute method. Set the CustomClassName configuration parameter in the Virus validation service transport.

The parameters for the execute method are

* url is the target url of the virus validation service.
* transactionData is the container that has all the pertinent information related to transaction in context. The file handlers for the attached files are in TransactionData attribute - 'com.pilotfish.mime.attachments'

Response will be a XML message stating the presence or absence of virus in the attachments. Below is the schema and sample XML file.

This method throws com.agencyport.substrate.exception.TransportFailedException when transport failure happens irrespective of underlying reason.

Schema

<xsd:schema xmlns:xsd=*"http://www.w3.org/2001/XMLSchema"* xmlns:tns=*"http://www.[Company].org/VirusValidation"* targetNamespace=*"http://www.[Company].org/VirusValidation"* elementFormDefault=*"qualified"*>

<xsd:simpleType name=*"Status\_Type"*>

<xsd:restriction base=*"xsd:string"*>

<xsd:enumeration value=*"No Virus"*></xsd:enumeration>

<xsd:enumeration value=*"Virus Detected"*></xsd:enumeration>

</xsd:restriction>

</xsd:simpleType>

<xsd:complexType name=*"VirusValidationResponse\_Type"*>

<xsd:sequence>

<xsd:element ref=*"tns:Status"* maxOccurs=*"1"* minOccurs=*"1"*>

</xsd:element>

</xsd:sequence>

</xsd:complexType>

<xsd:element name=*"VirusValidationResponse"* type=*"tns:VirusValidationResponse\_Type"*/>

<xsd:element name=*"Status"* type=*"tns:Status\_Type"*/>

</xsd:schema>

Sample XML File

<tns:VirusValidationResponse xmlns:tns=*"http://www.[Company].org/VirusValidation"* >

<tns:Status>No Virus</tns:Status>

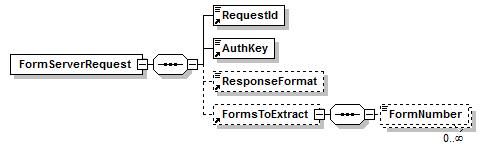
</tns:VirusValidationResponse>

## I. Turnstile Interface Structure

Turnstile service is a simple RESTful webservice that expects HTTPS request of content type multipart/form-data. Below are the details of the request and response.

### Request:

* **Request Schema:**



* **Sample Request:**

<FormServerRequest>

<RequestId>9ca66806-d31d-4afc-843c</RequestId>

<AuthKey>iOUJzYePC7zw4vsP0QdiyyYeJ8pJZNn/o62ogwEhR0</AuthKey>

<ResponseFormat>Standard</ResponseFormat>

<FormsToExtract>

<FormNumber>ACORD\_807</FormNumber>

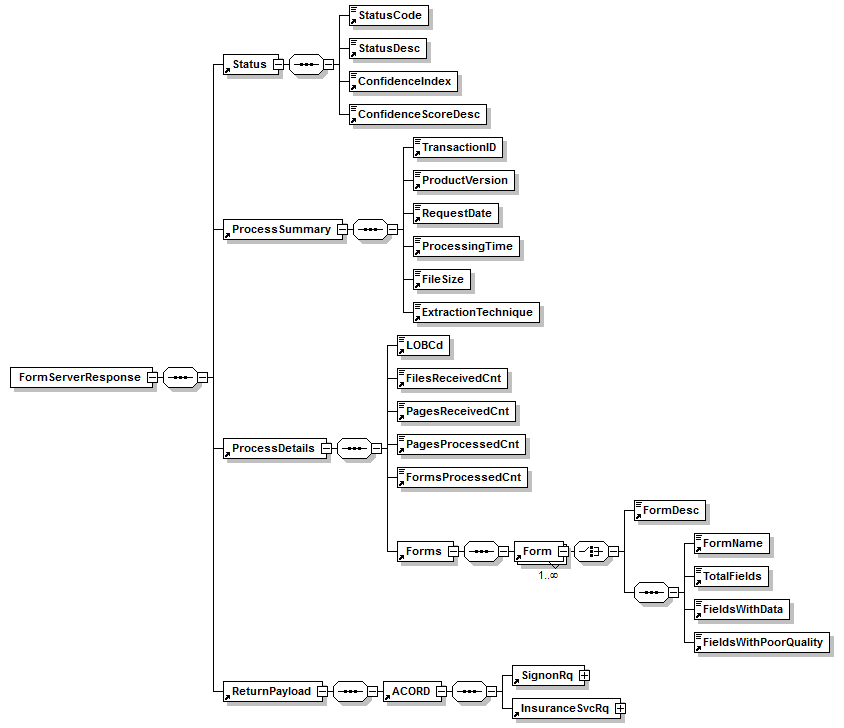
<FormNumber>ACORD\_825</FormNumber>

</FormsToExtract>

</FormServerRequest>

### Response:

Refer to the “form combinations“ specification published by turnstile to identity the number of InsuranceSvcRq’s and the ACORD message for each.

* **Response Schema:**
* **Sample Response:**

<FormServerResponse>

<Status>

<StatusCode>0</StatusCode>

<StatusDesc>Success</StatusDesc>

<ConfidenceIndex>A</ConfidenceIndex>

</Status>

<ProcessSummary>

<TransactionID>9ca66806-d31d-4afc-843c</TransactionID>

<ProductVersion>1.0.0.001</ProductVersion>

<RequestDate>2011-11-18</RequestDate>

<ProcessingTime>2 sec</ProcessingTime>

<FileSize>168892 bytes</FileSize>

<ExtractionTechnique>Text</ExtractionTechnique>

</ProcessSummary>

<ProcessDetails>

<LOBCd>AUTOB</LOBCd>

<FilesReceivedCnt>1</FilesReceivedCnt>

<PagesReceivedCnt>11</PagesReceivedCnt>

<PagesProcessedCnt>8</PagesProcessedCnt>

<FormsProcessedCnt>3</FormsProcessedCnt>

<Forms>

<Form>

<FormName>ACORD\_125\_2007\_07\_INS</FormName>

<TotalFields>449</TotalFields>

<FieldsWithData>72</FieldsWithData>

</Form>

<Form>

<FormName>ACORD\_137FL\_2006\_09\_INS</FormName>

<TotalFields>374</TotalFields>

<FieldsWithData>49</FieldsWithData>

</Form>

<Form>

<FormName>ACORD\_127\_2003\_08\_INS</FormName>

<TotalFields>606</TotalFields>

<FieldsWithData>139</FieldsWithData>

</Form>

</Forms>

</ProcessDetails>

<ReturnPayload>

<ACORD>

<SignonRq>

<SessKey>9ca66806-d31d-4afc-843c</SessKey>

<ClientDt>2011-11-18 07:21:50</ClientDt>

<CustLangPref>en-US</CustLangPref>

<ClientApp>

<Org>Agencyport</Org>

<Name>Turnstile</Name>

<Version>1.0</Version>

</ClientApp>

</SignonRq>

<InsuranceSvcRq>

<RqUID>629E417C-42F8-CA5F-83FB-C2595EA029D3</RqUID>

<CommlAutoPolicyQuoteInqRq>

<!-- ACORD DATA HERE -->

</CommlAutoPolicyQuoteInqRq>

</InsuranceSvcRq>

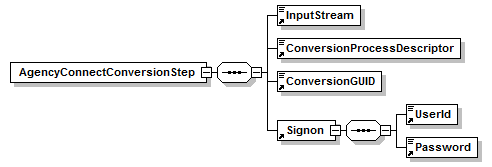
</ACORD>

</ReturnPayload>

</FormServerResponse>

## J. Upload Writer Interface Structure

### Request

* **Request Schema:**
* **Request Sample:**

<Connect5ConversionStep name="step6">

<InputStream>

<![CDATA[<ACORD><SignonRq>...</SignonRq><InsuranceSvcRq>...</InsuranceSvcRq></ACORD>]]>

</InputStream>

<ConversionProcessDescriptor>TODO</ConversionProcessDescriptor>

<ConversionGUID>E1E63BB2-2D36-A2D7-7B46-00A49EF71406</ConversionGUID>

<Signon>

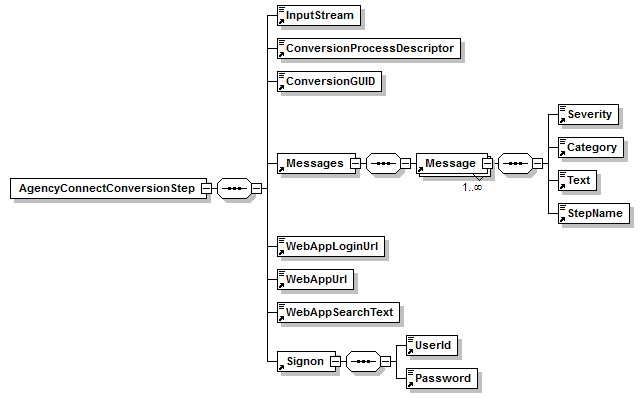
<UserId>agent</UserId>

<Password>password</Password>

</Signon>

</Connect5ConversionStep>

### Response

* **Response Schema:**
* **Response Sample:**

<Connect5ConversionStep name="step6">

<InputStream>

<![CDATA[<ACORD><SignonRq>...</SignonRq><InsuranceSvcRq>...</InsuranceSvcRq></ACORD>]]>

</InputStream>

<ConversionProcessDescriptor>TODO</ConversionProcessDescriptor>

<ConversionGUID>E1E63BB2-2D36-A2D7-7B46-00A49EF71406</ConversionGUID>

<Messages>

<Message>

<Severity>Warning</Severity>

<Category>Business</Category>

<Text>field key: 'commlAuto/generalInfo/controllingState/CommlPolicy.ControllingStateProvCd', validation category: 'VALIDATION\_CATEGORY\_ENTITY\_MISSING\_AND\_REQUIRED', error msg: 'Controlling

State.Controlling State is required', isKnockOutIssue: 'false', isHopeless: 'false', rule id: 'com.agencyport.fieldvalidation.validators.builtin.RequiredValidator' , page entity type:

FIELD\_ELEMENT, page entity compound key: 'commlAuto/generalInfo/controllingState/CommlPolicy.ControllingStateProvCd', page entity title: 'Controlling State', value: 'null', original value:

'null'</Text>

<StepName>upload writer:com.agencyport.commlAuto.upload.CAUploadDataManager</StepName>

</Message>

<Message>

<Severity>Informational</Severity>

<Category>System</Category>

<Text>Upload data saved to database: 1018</Text>

<StepName>step6</StepName>

</Message>

</Messages>

<WebAppLoginUrl> http://localhost:8080/portal/FrontServlet?TRANSACTION\_NAME=commlAuto&amp;PAGE\_NAME=generalInfo&amp;METHOD=Display&amp;FIRST\_TIME=true&amp;UPLOAD\_SITUATION=YES&amp;ConversionGUID=E1E63BB2-2D36-A2D7-7B46-00A49EF71406&amp;WORKITEMID=1018&amp;FAST\_FORWARD\_TO=generalInfo&amp;rand=0.3425117059273275</WebAppLoginUrl>

<WebAppUrl> http://localhost:8080/portal/FrontServlet?TRANSACTION\_NAME=commlAuto&amp;PAGE\_NAME=generalInfo&amp;METHOD=Display&amp;FIRST\_TIME=true&amp;UPLOAD\_SITUATION=YES&amp;ConversionGUID=E1E63BB2-2D36-A2D7-7B46-00A49EF71406&amp;WORKITEMID=1018&amp;FAST\_FORWARD\_TO=generalInfo&amp;rand=0.9307633101301971</WebAppUrl>

<WebAppSearchText>AgencyPortal\*</WebAppSearchText>

<Signon>

<UserId>agent</UserId>

<Password>password</Password>

</Signon>

</Connect5ConversionStep>

## K. File Attachment Interface

* Loosely coupled endpoint on the portal that will accept a multipart (contentType = ‘multipart/mixed’) request and save the received attachments into the workitem assistant or file attachment table.
* Connect5 Orchestration Layer will engage this route only when the [Company] Uploadwriter route succeeds in creating a workitem
* Connect5 Orchestration Layer will call the Portal FileAttachment servlet endpoint with a multipart request.
* Connect5 Orchestration Layer would pass workitem id and credentials (cookies) as part of the request.
* FileAttachmentManager will persist to work item assistant table or file attachment table based on the configuration.

Persisting to any other table is not in the scope and project teams would have to take up that work.

* Response from the portal just has the success/failure http status of the request.

Potential Changes to Portal:

* + Include the FileAttachmentServlet and FileAttachmentManager classes provided by Connect5 Orchestration Layer.
  + Make any changes if any to security filter to authenticate the request.
  + Configurable properties:

- turnstile.file.tablename : fileattachment or workitem assistant

-*LOBCD*.turnstile.file.document\_type: document type to be used for the attachments

Sequence diagram: Refer to 7.4.3

Sample Request:

POST <http://agencyportal.com/portal/FileAttachmentServlet>?work\_item\_id=11212 HTTP/1.1

Accept: \*/\*

Accept-Language: en\_US

Content-Type: multipart/mixed; boundary=--------------------12f949abd9f

Connection: Keep-Alive

Cache-Control: no-cache

Pragma: no-cache

User-Agent: Java/1.6.0\_18

Host: localhost:8888

Content-Length: 164216

----------------------12f949abd9f

Content-Type: application/pdf

%PDF-1.6

%ï¿½ï¿½ï¿½ï¿½

49 0 obj

<</Linearized 1/L 163549/O 51/E 87100/N 11/T 163141/H [ 525 295]>>

Endobj

----------------------12f949abd9f

Content-Type: application/pdf

%PDF-1.6

%ï¿½ï¿½ï¿½ï¿½

49 0 obj

<</Linearized 1/L 163549/O 51/E 87100/N 11/T 163141/H [ 525 295]>>

endobj