**ANIC Billing: ESB Design**

**Interface Specifications for PAS to BC Systems ~ Sync Policy**

**02/25/2021**

**Table of Contents**

[**Integration Overview**](#_heading=h.2s8eyo1) **4**

[**Document Control**](#_heading=h.17dp8vu) **4**

[**Contacts**](#_heading=h.3rdcrjn) **4**

[**Document History**](#_heading=h.26in1rg) **4**

[**Interface**](#_heading=h.lnxbz9) **5**

[**Purpose**](#_heading=h.35nkun2) **5**

[**Use Cases**](#_heading=h.1ksv4uv) **5**

[**Document Sign-Off**](#_heading=h.44sinio) **6**

[**Integration Details**](#_heading=h.z337ya) **6**

[**Basic Information**](#_heading=h.3j2qqm3) **6**

[**Data Transformation**](#_heading=h.4i7ojhp) **8**

[**Data Delivery/Retrieval Settings**](#_heading=h.1ci93xb) **8**

[**Processing Details**](#_heading=h.3whwml4) **9**

[**Pre-conditions**](#_heading=h.2bn6wsx) **9**

[**Post-conditions**](#_heading=h.qsh70q) **9**

[**Flow**](#_heading=h.3as4poj) **9**

[**Design Diagrams**](#_heading=h.49x2ik5) **14**

[**Sync Policy Diagram**](#_heading=h.2p2csry) **14**

[**Data Mapping Link/Sheet:**](#_heading=h.147n2zr) **14**

[**Design Assumptions & Decisions:**](#_heading=h.3o7alnk) **14**

[**Dependencies**](#_heading=h.23ckvvd) **15**

[**Security**](#_heading=h.ihv636) **15**

[**API Manager Configuration Parameters**](#_heading=h.32hioqz) **15**

[**Parameters**](#_heading=h.41mghml) **16**

[**Non-Functional Parameters**](#_heading=h.2grqrue) **16**

[**System Connection Details**](#_heading=h.3fwokq0) **17**

[**Error Handling**](#_heading=h.1v1yuxt) **17**

[**RAML Specification**](#_heading=h.4f1mdlm) **18**

[**Unit Testing & Sign-Offs**](#_heading=h.2u6wntf) **19**

[**Unit Testing Results**](#_heading=h.19c6y18) **19**

[**Client Sign Off**](#_heading=h.3tbugp1) **19**

[**Risk & Open Items**](#_heading=h.28h4qwu) **19**

[**Glossary**](#_heading=h.37m2jsg) **19**

[**Data Dictionary**](#_heading=h.1mrcu09) **20**

# 

# **Integration Overview**

|  |  |
| --- | --- |
| **Integration Overview** | |
| Integration Description | Sync Policy Transactions from PAS to BC |
| Application Area | ANIC Billing |
| Current And Future Architecture Scope | * Short Term   ☒ Long Term |

# **Document Control**

## **Contacts**

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Name** | **Email** | **Phone** |
| Technical Owner | Graham Baldeck | gbaldeck@copperpoint.com |  |
| Business Owner | TBD |  |  |
| Engagement Director - PwC | Dhirendra Tiwari | dhirendra.tiwari@pwc.com | 980-226-3147 |
| Technical Architect | Damo Seella | damodar.seella@pwc.com | 337-529-7078 |

## **Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version No.** | **Date**  **(MM/DD/YYYY)** | **Version Description** | **Author** |
| 1.0 | 01/15/2021 | Document Created & Initial Draft | Kobi Galvis - PwC |
| 1.2 | 01/29/2021 | Document edits | Kobi Galvis - PwC |
| 1.3 | 02/17/2021 | Document updates | Kobi Galvis - PwC |
| 1.4 | 02/25/2021 | Final edits before conditional approval | Kobi Galvis - PwC |
| 1.5 | 03/01/2021 | Added replay mechanism description & reinstate flow description edit | Kobi Galvis - PwC |
| 1.6 | 03/02/2021 | Added the Transaction Key & Edited the replay mechanism description | Kobi Galvis - PwC |
| 1.7 | 03/10/2021 | Edited RAML names | Kobi Galvis - PwC |
| 1.8 | 03/18/2021 | Updated vCores | Kobi Galvis - PwC |

# **Interface**

## **Purpose**

To sync policy transactions from the Policy Administration System (PAS) to the Billing Center (BC).

## **Use Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case #** | **API to be Used #** | **Transport Method** | **Description** |
| 1. New Business | 5 | TLS1.2, HTTPS | New business created in the Billing Center |
| 2. Policy Change | 5 | TLS1.2, HTTPS | Changes of the policy in the Billing Center |
| 3. Renewal | 5 | TLS1.2, HTTPS | Renewal of policy in the Billing Center |
| 4. New Renewal | 5 | TLS1.2, HTTPS | New renewal in the Billing Center |
| 5. Cancellation | 5 | TLS1.2, HTTPS | Cancellation of policy in the Billing Center |
| 6. Premium Reporting | 5 | TLS1.2, HTTPS | Premium reporting of policy in the Billing Center |
| 7. Final Audit | 5 | TLS1.2, HTTPS | Final audit of policy in the Billing Center |

# **Document Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Name / Signature** | **Title** | **Date** |
| **Technology SME** | Naveen Kammula | Director Application Development |  |
| **Business SME** | TBD |  |  |

# 

# **Integration Details**

## **Basic Information**

|  |  |  |
| --- | --- | --- |
| **Basic Settings** | | |
| **Integration System Name** | | MuleSoft |
| **Real Time/ Batch** | | * Real time * Batch |
| **Integration Direction** | | * Inbound * Outbound |
| **Batch Mode** | | * Delta (Changes Only) * Full Data * N/A |
| **API/Protocol** | | * SOAP * REST * Bulk * Connector |
| **Data Source Type** | | * Custom Report * Rest URL * Web Service * File |
| **Data Source Name** | | ANIC PAS |
| **Input Parameters** | | Please refer to *“AB-639 - CopperPoint Insurance - ESB Design - Sync Policy - Mapping Document”* |
| **Output Parameters** | | Fore detailed output parameters, please refer to *“AB-639 - CopperPoint Insurance - ESB Design - Sync Policy Mapping Document”* |
| **Integration Schedule** | **Frequency** | * Ad-hoc * Daily * Weekly * Monthly * Semi-Monthly * Annually * Bi-Weekly * Other   ESB will be supporting any ad-hoc requests.  As per PAS requirements, it will be a daily batch which will invoke ESB API integration. |
| **Day of the Week /**  **Date & Month** | N/A (Real time) |
| **Time** | N/A (Real time) |
| **Expected Data Volume** | | 3-5 MB per day |
| **payload type** | | JSON / XML |
| **Caching Enabled** | | N/A |
| **Data Classification** | | PII |

## 

## **Data Transformation**

|  |  |
| --- | --- |
| **Data Transformation** | |
| **Transformation Type** | * Predefined Transformation * Custom Transformation * None |
| **Transformation Name** | Please refer to *“AB-639 - CopperPoint Insurance - ESB Design - Sync Policy Mapping Document”* for transformation |

## 

## **Data Delivery/Retrieval Settings**

|  |  |
| --- | --- |
| **Data Delivery/Retrieval Settings** | |
| **Delivery/Retrieval Method** | * Email Attachment * FTP with PGP Encryption * SFTP over SSH * SFTP with Password * HTTP(S) |
| **Data Format** | * XML * CSV * Text – Fixed Width * Text – Pipe-Delimited   ☒ JSON |

# **Processing Details**

## **Pre-conditions**

* + 1. All APIs are built and tested
    2. Sync Account has been completed

## **Post-conditions**

* + 1. N/A

## **Flow**

1. The ANIC PAS sends the Policy Transaction data to the exp-pas-policies API within the MuleSoft ESB
   1. The exp-pas-policies API sorts the data by transaction number and publishes it to Anypoint MQ
   2. In sequence, the int-dynamodb-utilities API calls a POST function to create a parent record in DynamoDB
2. The prc-policies-orchestration API consumes the message from Anypoint MQ and executes the “can proceed check” logic
   1. The Transaction Key should be set as -- Source System '-' concat Target System concat '-' concat Transaction Type concat '-' concat Transaction Key
   2. The prc-policies-orchestration API acquires the transaction status by calling a GET function from the int-dynamodb-utilities API to request if the transaction can proceed
   3. If the transaction can not proceed to be processed, send this information to Anypoint MQ which will then update DynamoDB through the int-dynamodb-utilities API
   4. If the transaction can proceed to be processed, the prc-policy-center API updates the status to “In Process”
   5. Then the prc-policy-center API calls a GET function to request LookUp data from the int-dynamodb-utilities API
3. From here, we continue from where the prc-policies-orchestration API in the Sync Account diagram merges into the Sync Policy flow
   1. Flow found in *AB-89 - CopperPoint Insurance - ESB Design - Sync Accounts Sequence Diagram*
4. Now, the prc-policies-orchestration API determines the policy transaction type
   1. If the Policy Transaction type is a new business:
      1. The prc-policies-orchestration API searches for the policy in the Billing Center through the prc-billing-center API by calling a GET function to retrieve the policy number
      2. If the policy does not exist, the prc-policies-orchestration API creates the policy in the Billing Center through the prc-billing-center API calling a POST function to issue a new policy
      3. If the policy does exist, the prc-policies-orchestration API determines if the policy was canceled or not
         1. If the policy was canceled, the prc-policies-orchestration API reinstates the policy through the prc-billing-center API by calling a PATCH function in order to reinstate the policy
            1. Also, publish the complete transaction to Anypoint MQ and update the status to “complete”
         2. If the policy was not canceled, throw an error and follow these three steps:
            1. If there are any failed transactions, stop processing the next transaction in the same array
            2. Publish the error transaction to Anypoint MQ
            3. Update the status to “Error”
      4. The prc-billing-center API has many functionalities: it calls GET functions to retrieve the policy through the policy number, POST functions to issue new policies, and PATCH functions to update policy renewals, changes, cancelations, reinstatements, final audits, premium reports, and rewrites
         1. The sys-gwbc-policies API calls a GET function to retrieve the policy number so that the Billing Center can search for the policy
         2. Next, the sys-gwbc-policies API calls a POST function so that the Billing Center can issue a new policy
         3. Continuing, the sys-gwbc-policies API calls a PATCH function so that the Billing Center can renew the policy
         4. Then, the sys-gwbc-policies API calls a PATCH function so that the Billing Center can change the policy
         5. If necessary, the sys-gwbc-policies API calls a PATCH function so that the Billing Center can cancel the policy
         6. Next, the sys-gwbc-policies API calls a PATCH function so that the Billing Center could reinstate the policy
         7. Continuing, the sys-gwbc-policies API calls a PATCH function so that the Billing Center can provide a final audit of the policy
         8. Then, the sys-gwbc-policies API calls a PATCH function so that the Billing Center could provide a premium report of the policy
         9. Finally, the sys-gwbc-policies API calls a PATCH function so that the Billing Center would be able to rewrite the Policy
   2. If the Policy Transaction type is a renewal:
      1. The prc-policies-orchestration API searches for the prior policy in the Billing Center through the prc-billing-center API by calling a GET function to retrieve the policy number
      2. If the prior policy does not exist, call the New Business Flow from step 4a and follow those steps
      3. If the prior policy does exist, the prc-policies-orchestration API searches for the current policy in the Billing Center through the prc-billing-center API by calling a GET function to retrieve the policy number
      4. If the current policy does not exist, the prc-policies-orchestration API renews the policy through the prc-billing-center API by calling a PATCH function in order to renew the policy
         1. Now, follow the steps from 4.a.iv.
      5. If the current policy does exist, the prc-policies-orchestration API determines if the policy has been cancelled or not
         1. If the current policy has not been cancelled, throw an error and follow these three steps:
            1. If there are any failed transactions, stop processing the next transaction in the same array
            2. Publish the error transaction to Anypoint MQ
            3. Update the status to “Error”
         2. If the current policy has been cancelled, the prc-policies-orchestration API reinstates the policy through the prc-billing-center API by calling a PATCH function in order to reinstate the policy
            1. Now, follow the steps from 4.a.iv.
            2. Also, publish the complete transaction to Anypoint MQ and update the status to “complete”
   3. If the Policy Transaction type is a policy change:
      1. The prc-policies-orchestration API searches for the policy in the Billing Center through the prc-billing-center API by calling a GET function to retrieve the policy number
      2. If the policy does exist, the prc-policies-orchestration API changes the policy through the prc-billing-center API by calling a PATCH function in order to change the policy
         1. Now, follow the steps from 4.a.iv.
         2. Also, publish the complete transaction to Anypoint MQ and update the status to “complete”
      3. If the policy does not exist, throw an error and follow these three steps:
         1. If there are any failed transactions, stop processing the next transaction in the same array
         2. Publish the error transaction to Anypoint MQ
         3. Update the status to “Error”
   4. If the Policy Transaction type is a policy cancellation:
      1. The prc-policies-orchestration API searches for the policy in the Billing Center through the prc-billing-center API by calling a GET function to retrieve the policy number
      2. If the policy does exist, the prc-policies-orchestration API cancels the policy through the prc-billing-center API by calling a PATCH function in order to cancel the policy
         1. Now, follow the steps from 4.a.iv.
         2. Also, publish the complete transaction to Anypoint MQ and update the status to “complete”
      3. If the policy does not exist, throw an error and follow these three steps:
         1. If there are any failed transactions, stop processing the next transaction in the same array
         2. Publish the error transaction to Anypoint MQ
         3. Update the status to “Error”
   5. If the Policy Transaction type is a policy reinstatement:
      1. The prc-policies-orchestration API searches for the policy in the Billing Center through the prc-billing-center API by calling a GET function to retrieve the policy number
      2. If the policy does exist and the status is “cancelled,” the prc-policies-orchestration API reinstates the policy through the prc-billing-center API by calling a PATCH function in order to reinstate the policy
         1. Now, follow the steps from 4.a.iv.
         2. Also, publish the complete transaction to Anypoint MQ and update the status to “complete”
      3. If the policy does not exist, throw an error and follow these three steps:
         1. If there are any failed transactions, stop processing the next transaction in the same array
         2. Publish the error transaction to Anypoint MQ
         3. Update the status to “Error”
   6. If the Policy Transaction type is a policy audit:
      1. The prc-policies-orchestration API searches for the policy in the Billing Center through the prc-billing-center API by calling a GET function to retrieve the policy number
      2. If the policy does exist, the prc-policies-orchestration API does a final audit of the policy through the prc-billing-center API by calling a PATCH function in order to provide the final audit
         1. Now, follow the steps from 4.a.iv.
         2. Also, publish the complete transaction to Anypoint MQ and update the status to “complete”
      3. If the policy does not exist, throw an error and follow these three steps:
         1. If there are any failed transactions, stop processing the next transaction in the same array
         2. Publish the error transaction to Anypoint MQ
         3. Update the status to “Error”
   7. If the Policy Transaction type is a policy premium report:
      1. The prc-policies-orchestration API searches for the policy in the Billing Center through the prc-billing-center API by calling a GET function to retrieve the policy number
      2. If the policy does exist, the prc-policies-orchestration API provides a premium report of the policy through the prc-billing-center API by calling a PATCH function in order to provide the premium report
         1. Now, follow the steps from 4.a.iv.
         2. Also, publish the complete transaction to Anypoint MQ and update the status to “complete”
      3. If the policy does not exist, throw an error and follow these three steps:
         1. If there are any failed transactions, stop processing the next transaction in the same array
         2. Publish the error transaction to Anypoint MQ
         3. Update the status to “Error”

## Replay Mechanism Flow

The replay mechanism allows operations to recover and replay failed transactions.

1. The correlation-id needs to be sent to the int-replay API
2. Next, make a call to DynamoDB to get the parent payload
3. Then, the int-replay API will publish the message to the corresponding MQ
4. Now, the message will be processed as a regular transaction

# **Design Diagrams**

# **Sync Policy Diagram**

# **Data Mapping Link/Sheet:**

Please refer to *“AB-639 - CopperPoint Insurance - ESB Design - Sync Policy Mapping Document”*

# **Design Assumptions & Decisions:**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Assumption/Decision** | **Description** |
| 1 | Assumption | Leverage the standard error handling & logging frameworks as agreed upon with CopperPoint team |
| 2 | Assumption | ANIC PAS will call syncPolicy API; 1 call for each policy (1 call can have multiple transactions in each policy) |
| 3 | Assumption | In case of an error in the API call, ANIC PAS will need to resend all the transactions included in the request |
| 4 | Assumption | PaymentPlan will have a mapping association and IDs are provided to match |
| 5 | Decision | API orchestration will be done at Mulesoft layer |
| 6 | Decision | Data transformation will be done in Mulesoft layer |
| 7 | Decision | Duplicate Check for Policy Transactions |

# **Dependencies**

|  |  |  |
| --- | --- | --- |
| **Dependency #** | **Description** | **Owner** |
| 1 | PolicyCenter API development to support the integration | Core Guidewire team |
| 2 | BillingCenter API development to support the integration | Core Guidewire team |

# **Security**

## **API Manager Configuration Parameters**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Name of the Policy** | **Values** |
| 1 | Client Id Enforcement | Client Id:  Client Secret: |
| 2 | Experience Layer: All incoming requests will be authenticated by Basic Authentication: LDAP Policy | Username:  Password: |
| 3 | All APIs should be exposed to HTTPS | Client Certificates |

## 

# **Parameters**

## Non-Functional Parameters

## 

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Name** | **Description** | **Values** |
| 1 | Maximum throughput | Provide highest expected volume in the shortest time frame | 500-600 transactions per day, which means approximately 500 API calls |
| 2 | Caching Enabled | Any in-memory Cache/Object-Store defined for the integration? | N/A |
| 3 | vCores | No of vCores & size of vCores used for the integration | 0.1 per app |
| 4 | Data Classification | Describe the data classification assigned to the interface/payload | PII |
| 5 | Payload Size | Define the size & Maximum depth/nesting | 5 mb per day (approx. 600 payload transactions) |
| 6 | Logging | Any restriction in logging PII/PCI data | TBD |
| 7 | Anypoint Alerts | Review the system alerts available in Anypoint Platform & decide which one to implement | Generic Monitoring Alerts in Anypoint Platform |

## System Connection Details

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **System Name** | **Connection Parameters** | **Details** |
| 1 | Policy Center (PC) | <https://gwrivwas026.copperpoint.com:8743/pc/ws/gw/webservice/pc/pc1000/account/AccountAPI?WSDL> | **Operations:**  addExternalAccount  updateExternalAccount  getContactByAccountNumber  **Note:** INT2 Environment which will be used for testing |
| 2 | Billing Center (BC) | <https://gwrivwas026.copperpoint.com:8643/bc/ws/gw/webservice/policycenter/bc1000/BillingAPI?WSDL> | **Operations:**  createAccount  SearchForAccounts  updateAccount  **Note:** INT2 Environment which will be used for testing |

# **Error Handling**

Please note that this is an initial list. As we continue discussions, this list will be updated.

|  |  |
| --- | --- |
| **Error Description** | **Action** |
| Process API: stored failed transaction in DB | Send response back to PAS support |
| Validation Error (Custom error code) | Email notification to corresponding teams |
| Process Error (Custom error code) | Email notification to corresponding teams |

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Return Values/Conditions** | **Error Code** | **Error Message** |
| Input Request is Not Valid | Input validation failed: (1) input field cannot be empty (2) input fields exceeds size limit | 400 | Bad Request - for invalid request (badly formatted request, payload or parameter) or validation failed. |
| Cloudhub Worker Report Errors | (1) Dataweave expression error  (2) Internal server error  (3) Maximum stream size exhausted error | 500 | Internal Server Error. Contact system administrator for more details. |

# **RAML Specification**

PAS Policies XAPI

* [https://anypoint.mulesoft.com/exchange/d3d83e42-4755-45b8-96b1-e79da744fb32/policies-pas-xapi/](https://anypoint.mulesoft.com/exchange/d3d83e42-4755-45b8-96b1-e79da744fb32/policies-pas-xapi/minor/1.0/pages/home/)

Policy Center PAPI

* <https://anypoint.mulesoft.com/exchange/d3d83e42-4755-45b8-96b1-e79da744fb32/policy-center-papi/>

prc-policies-orchestration

* As this API integrates with Anypoint MQ, there will not be an associated RAML

Billing Center PAPI

* <https://anypoint.mulesoft.com/exchange/d3d83e42-4755-45b8-96b1-e79da744fb32/billing-center-papi/>

GWBC Policies SAPI

* <https://anypoint.mulesoft.com/exchange/d3d83e42-4755-45b8-96b1-e79da744fb32/policy-bc-sapi/>

DynamoDB Utilities IAPI

* <https://anypoint.mulesoft.com/exchange/d3d83e42-4755-45b8-96b1-e79da744fb32/dynamo-db-utilities-iapi/>

Replay IAPI

* <https://anypoint.mulesoft.com/exchange/d3d83e42-4755-45b8-96b1-e79da744fb32/replay-iapi/>

# **Unit Testing & Sign-Offs**

## **Unit Testing Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case #** | **Test Case** | **Test Data** | **Expected Result** | **Actual Result** |
|  |  |  |  |  |

## **Client Sign Off**

*Embed the sign-off mail from the vendor in PDF format.*

# **Risk & Open Items**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Item** | **Owner** | **Expected Closure Date** |
|  |  |  |  |

# 

# **Glossary**

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| HTTPS | Hypertext Transfer Protocol Secure |
| JSON | JavaScript Object Notation |
| REST | Representational State Transfer |
| SFTP | Secured File Transfer Protocol |
| SOAP | Simple Object Access Protocol |
| SSH | Secure Shell |
| XML | Extensible Markup Language |
| Predefined Transformation | Data mapping between source fields and target fields without any transformation logic |
| Custom Transformation | Data mapping between source fields and target fields including any transformation logic |

# **Data Dictionary**

\*Disclaimer: some of the fields left blank do not exist in Guidewire yet\*

|  |  |
| --- | --- |
| **Data / Acronym** | **Description** |
| Transaction Type |  |
| Transaction # | The number for this transaction |
| Transaction Date | The date of the transaction |
| Source |  |
| policyInfo | |
| Account # |  |
| Policy # | The policy number for this policy period |
| Term Number | The number indicates the term of the policy period, starts with 1 and then incremented by 1 for every renewal or rewrite |
| Term Confirmed | True if the policy period term is confirmed by the PAS (default true). |
| Previous Policy # | Lob Code |
| Model Date | Date corresponding to the model number |
| Effective Date | Derived property returning java.util.Date |
| Expiration Date | Expiration Date |
| Transaction Effective Date | It will depend of if the policy is reinstated with a lapse or not |
| Description |  |
| Cancel Source |  |
| Cancel Reason | The reason a policy was cancelled |
| Cancel Method |  |
| Deviation Factor |  |
| Audit Status | Audit Status |
| Audit Method | Audit Method |
| Audit Period Start Date |  |
| Audit Period End Date |  |
| Payment Received | Whether the Policy Administration System had received payment for the premium report at the time of this billing instruction |
| Reporting Method Ext | Reporting Method |
| Product Code |  |
| Assigned Risk Indicator | Indicates whether this policy period is written as part of a mandatory program for covering the worst risks |
| NCCI Binder # |  |
| Jurisdiction | The state for the installment fee |
| UW Company | The UW company writing this policy period |
| Eligible For Full Pay Discount | Whether the policy is eligible for a full pay discount |
| Require Final Audit | Derived property returning boolean |
| Billing Method | How to determine the default payer for this PolicyPeriod |
| Overriding Payer Account Number | Account to use as the payer for new charges and items on this PolicyPeriod |
| Payment Plan | The payment plan in use on this policy period |
| Producer Code | The related producer code |
| Currency | The currency used by this entity's silo. |
| policyInfo/primaryNamedInsured | |
| Public ID | ID or primary key of the row in the external system to which this row is mapped |
| Address Book UID | Represents the ID of the associated object in Address Book. Null if the object is not linked to Address Book. |
| Type |  |
| Name | This contact's name. |
| Address 1 | Primary address associated with the contact. First line of mailing address. |
| Address 2 | Secondary addresses associated with the contact. Second line of mailing address. |
| City | City |
| State | State |
| ZIP Code | Zip |
| Country | Country |
| Phone | Primary phone number type for the contact. |
| Email | Primary email address associated with the contact. |
| FEIN |  |
| policyInfo/dba | |
| Type |  |
| name | This contact's name. |
| Address 1 | Primary address associated with the contact. First line of mailing address. |
| Address 2 | Secondary addresses associated with the contact. Second line of mailing address. |
| City | City |
| Insured State | State |
| Zip Code | Zip |
| country | Country |
| DBA Phone |  |
| Insured Email |  |
| DBA FEIN |  |
| policyInfo/additionalNamedInsured (unbounded) | |
| Public ID | ID or primary key of the row in the external system to which this row is mapped |
| Address Book UID | Represents the ID of the associated object in Address Book. Null if the object is not linked to Address Book. |
| Type |  |
| Name | This contact's name. |
| Address 1 | Primary address associated with the contact. First line of mailing address. |
| Address 2 | Secondary addresses associated with the contact. Second line of mailing address. |
| City | City |
| State | State |
| ZIP Code | Zip |
| Country | Country |
| Phone |  |
| Email |  |
| FEIN |  |
| policyInfo/chargeInfo (unbounded) | |
| Charge Name |  |
| Amount | The amount of the charge |
| Group | A String that logically groups this charge, e.g., 'collision', 'comprehensive'; this is unrelated to the charge's ChargePattern |
| WrittenDate | The date the charge was sent by PAS |
| Commission Rate |  |