SIDES

NASWA State Information Data Exchange System

In Partnership with USDOL

DEVELOPER GUIDE

PART C – DEVELOPMENT STEPS

Warning: This is a redacted version of the SIDES Developer Guide and is NOT the latest version. For development, log into the Members Site and obtain the latest version.

Version: 2.3

Date: February 22, 2017

Revision History

Date	Version	Description	Author
8/20/2010	1.0	Version 1.0	SIDES Team
9/27/2010	1.1	Updated: Part C – Development Steps. Added discussion on Combined.xsd to document. See Sections 3.3, 3.3.1.4, and 4.6.1. Enhanced the description of a GUID. See Section 4.2.2. Enhanced discussion on certification test data files to indicate that connectors may need to edit certification test data files. See Section 8.1.1.1.1.	SIDES Team
12/09/2010	1.2	Updated: Part C – Development Steps. Under section 9.1, Common Mistakes, two new sections were added. Section 9.1.7 clarifies he interpretation of numeric fields used to store money values. Section 9.1.8 provides guidance on populating the StateEmployerAccountNbr field so employers and TPAs can use this data to look up employer information in their automated systems.	SIDES Team
4/7/2011	2.0	Added Earnings Verification requirements, modified requirements to be specific to their exchange, added requirements from CCB#9 (C-2.3.3.1-4, 2.3.3.1-5, and 2.3.3.1-6, added Model Connector.	SIDES Team
5/20/2011	2.1	Updated Part C. Added Employer Model Connector; Jax-WS Model Connector and .Net Model Connector; Corrected SOAP header information relating to the Earnings Verification exchange	SIDES Team

Date	Version	Description	Author
11/17/2011	2.2	Updated Part B. Added section C-2.10 SEW and added in the requirement to specify the size of the SEW custom logo to C-2.10.1. Identified requirements met through the use of the SIDES Model Connector.	SIDES Team
2/22/2017	2.3	Updated SIDES Logo	SIDES Team

TABLE OF CONTENTS

1 INTRODUCTION

This document is the final part of the comprehensive Developer Guide package. Part C contains technical guidance to assist you in building the connector to communicate with the **Central Broker.** Please note that it is very important that you have addressed all of the issues outlined in Part B – Connector Requirements before you start with this part.

The document is composed of the following:

- **Initial Setup Instructions**
- **A – Collect and Arrange Data**
  - Data Requirements
  - XML (includes samples)
  - XSD
  - File Size
  - Business Rules
- **B - Preparing the Message**
  - Messaging Overview – Post, Pull, and Push
  - Messaging Concepts
  - SOAP Customer Headers
  - SOAP Payload
  - SOAP Actions
  - WSDL
- **C - Securing the Message**
  - **REDACTED**
- **D - Sending the Message**
  - Sending a message
  - Sample SOAP Message
  - Acknowledgements
  - Non-broker Returns
- **E - Testing the Connector Software**
  - Connector Responsibility
  - Tools
    - Model Connectors
    - Business Rule Processor Tool (BRPT)
- **F – Certifying the Connector Software**
  - Step 1 - Download test suite
  - Step 2 - Conduct preliminary connector certification testing
  - Step 3 - Submit the Spreadsheet

    - Step 4 – Conduct Final Connector Certification Test
- **Common Mistakes, Things to Remember, Key Development Pitfalls**
- **List of Tables**

Each section contains specific information and examples of how to ensure you are able to successfully execute each process.

For the connector to interface with the **SIDES** **Central Broker**, the connector must obtain their data from their backend system. This initial step is outside of the scope of the developer guide. Once you obtain the data you need to put it into XML as prescribed by the **SIDES** XSDs. The development steps documented herein will guide your development of a **SIDES** compliant connector.

2 INITIAL SETUP INSTRUCTIONS

To connect to the SIDES **Central Broker**, the connector needs to obtain and configure the appropriate URL to access the SIDES test and SIDES production environment. Public keys must be exchanged between the connector and the **Central Broker**. The SIDES **Broker Administrator** must set up the connector within the SIDES Admin Site.

2.1 URLs

The URLs used by the connectors for all messaging with the **Central Broker** are:

- **REDACTED** [Separation Information production]
- **REDACTED** [Separation Information test]
- **REDACTED** [Earnings Verification production]
- **REDACTED** [Earnings Verification test]

2.2 Public Key Exchange - REDACTED

REDACTED

3 A – COLLECT AND ARRANGE DATA

Prior to collecting and arranging data, the connector must perform data analysis and mapping between their backend system, the Separation Information Exchange Format, and the Earnings Verification Exchange Format (hereafter known only as Exchange Format). Once the analysis is complete, the connector must extract data from their backend system and generate an XML file, which will be packaged and delivered to the **Central Broker**. This section: describes **SIDES** data requirements; introduces XML, XSDs; defines the date data type used by **SIDES**; introduces how attachments are transmitted to the **Central Broker** using Message Transmission Optimization Mechanism (MTOM); explains the backfilled data; specifies the **SIDES** file size; and articulates the business rules that must be followed to participate in **SIDES**.

3.1 Data Requirements

The data required for the requests and responses is a set of predetermined data elements, each having its own individual requirements along with its interaction with other elements. The request and response data elements (the standard format) can be found in the **SIDES** Implementation Guide. This Development Steps guide contains a technical discussion of the information provided for each field in the Separation Information Exchange Format and the Earnings Verification Exchange Format.

> ***Note***: For the latest version of Separation Information Exchange Format or Earnings Verification Exchange Format, please visit the **SIDES** Website (http://sides.itsc.org), and obtain the SIDES Implementation Guide, which contains the Separation Information Exchange Format and Earnings Verification Exchange Format.

There are two tables in the Exchange Formats. The first table outlines the data requirements for the Request sent by a State. The second table outlines the data requirements for the Response sent by an employer or TPA in response to the Request.

Table 1 describes the information specified for each request and response data element listed in the Exchange Format.

The responsibility of the development team for a **SIDES** connector is to (a) ensure that, given the information in the Exchange Format, the connecting system can populate all of this data with the development team’s current back-end systems or future planned ones and (b) ensure their client can pass all validations and business rules before they deploy to production.

Table 1 – Exchange Format

Column Name	Definition
Seq. Number	The “Sequence Number” is the identifier for the data element. It helps identify which data element is being discussed.
Data Element Name	The “Data Element Name” is the name of the data element.
Data Element Description	The “Data Element Description” is the description of the data element.

Column Name	Definition
Type and/or Format	The “Type and/or Format” is the format of the data element (i.e. character, a numeric, a date or a base64Binary data field).
Field Size	The “Field Size” is the size of the data element. Depending on what type of element it is, it may be a maximum size (2000 character string), or it may be the exact size (10 character date).
Field Required / Optional	“Field Required / Optional” tells whether the field must have an answer for the data element. There are three types of data elements: Required, Optional, and Conditional. • Required means that an answer must be filled in. • Optional means that an answer should only be filled in if it makes sense for the state, employer, or TPA to fill in that answer. • Conditional means that an answer may be required or optional depending on the answers to other data elements. (See Business Rules for more information.)
Business Rules	“Business Rules” are the directives that must be met when filling in this data element. These can include the reasons that will make a conditional element required or restrictions on a date field.
Validation	“Validation” is a check on the data to make sure that the data is in a consistent form and can be readable by all participants.
Comments/Notes	“Comments” are general comments on the data element.
Values	The “Values” column contains all the values that a data element may have for those data elements that are restricted to a set of values.

3.2 XML

XML was selected as the medium to implement the **SIDES** Exchange Format specification for the transfer of the data between states, employers, and TPAs. XML allows the data to be defined in a clear and concise manner.

State, employer, and TPA connectors must implement clients that can correctly communicate using the **SIDES** messaging specification, per the current Requirements Baseline, system design, and as elucidated in this guide.

3.2.1 Sample State Separation Request

```
<StateSeparationRequest>
      <StateRequestRecordGUID>50000000000000000000000000003364</StateRequestR
ecordGUID>
      <SSN>000128475</SSN>
      <ClaimEffectiveDate>2009-06-04</ClaimEffectiveDate>
      <ClaimNumber>378620</ClaimNumber>
      <StateEmployerAccountNbr>0064560</StateEmployerAccountNbr>
      <EmployerName>JC PENNEY COMPANY INC</EmployerName>
      <FEIN>794741844</FEIN>
      <TypeofEmployerCode>3</TypeofEmployerCode>
      <TypeofClaimCode>1</TypeofClaimCode>
      <BenefitYearBeginDate>2009-06-04</BenefitYearBeginDate>
      <RequestingStateAbbreviation>UT</RequestingStateAbbreviation>
      <UIOfficePhone>8015264400</UIOfficePhone>
      <UIOfficeFax>8015269394</UIOfficeFax>
      <ClaimantLastName>FUKBPHM</ClaimantLastName>
      <ClaimantFirstName>ELSA</ClaimantFirstName>
      <WagesWeeksNeededCode>NA</WagesWeeksNeededCode>
      <ClaimantSepReasonCode>99</ClaimantSepReasonCode>
      <RequestDate>2009-06-07</RequestDate>
      <ResponseDueDate>2009-06-17</ResponseDueDate>
      <FormNumber>606C</FormNumber>
 </StateSeparationRequest>
```

3.2.2 Sample Employer/TPA Separation Response

```
<EmployerTPASeparationResponse>
      <StateRequestRecordGUID>00000000000000000000000000003364</StateRequestRecordGUID>
      <BrokerRecordTransactionNumber>2001753</BrokerRecordTransactionNumber>
      <SSN>000128475</SSN>
      <ClaimEffectiveDate>2009-06-04</ClaimEffectiveDate>
      <ClaimNumber>378620</ClaimNumber>
      <StateEmployerAccountNbr>0064560</StateEmployerAccountNbr>
      <ClaimantNameWorkedAsForEmployer>Elsa LKJFGRE2</ClaimantNameWorkedAsForEmployer>
      <ClaimantJobTitle>Customer Service Associate</ClaimantJobTitle>
      <SeasonalEmploymentInd>N</SeasonalEmploymentInd>
      <EmployerReportedClaimantFirstDayofWork>2005-10-11</EmployerReportedClaimantFirstDayofWork>
      <EmployerReportedClaimantLastDayofWork>2008-10-14</EmployerReportedClaimantLastDayofWork>
      <EffectiveSeparationDate>2008-10-14</EffectiveSeparationDate>
      <TotalEarnedWagesNeededInd>2</TotalEarnedWagesNeededInd>
      <TotalEarnedWages>0</TotalEarnedWages>
      <TotalWeeksWorked>0</TotalWeeksWorked>
      <WagesEarnedAfterClaimEffectiveDate>0</WagesEarnedAfterClaimEffectiveDate>
      <NumberOfHoursWorkedAfterClaimEffectiveDate>0</NumberOfHoursWorkedAfterClaimEffectiveDate>
      <AverageWeeklyWage>0</AverageWeeklyWage>
      <EmployerSepReasonCode>6</EmployerSepReasonCode>
      <VoluntarySepReasonCode>1</VoluntarySepReasonCode>
      <ClaimantActionsToAvoidQuitInd>N</ClaimantActionsToAvoidQuitInd>
      <ContinuingWorkAvailableInd>Y</ContinuingWorkAvailableInd>
      <VoluntarySepReasonComments>The claimant quit without giving JCPenney a reason.</VoluntarySepReasonComments>
      <PreparerTypeCode>E</PreparerTypeCode>
      <PreparerTelephoneNumberPlusExt>9724312108</PreparerTelephoneNumberPlusExt>
      <PreparerContactName>Jay Johns</PreparerContactName>
      <PreparerTitle>Project Manager</PreparerTitle>
      <PreparerFaxNbr>9725312108</PreparerFaxNbr>
      <PreparerEmailAddress>sample@jcpenney.com</PreparerEmailAddress>
</EmployerTPASeparationResponse>
```

3.3 Sample State Earnings Verification Request

```
<StateEarningsVerificationRequestCollection xsi:schemaLocation="https://
REDACTED schemas EarningsVerificationRequest.xsd" xmlns="https://
REDACTED /schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <StateEarningsVerificationRequest>
      <StateEarningsVerificationRequestRecordGUID>AAA53000000000000000000000000000003</StateEarningsVerificationRequestRecordGUID>
      <RequestingStateAbbreviation>ST</RequestingStateAbbreviation>
      <UIOfficeName>Office Name</UIOfficeName>
      <UIOfficePhone>5555555555</UIOfficePhone>
      <UIOfficeFax>5555555554</UIOfficeFax>
      <UIOfficeEmailAddress>james.madison@state.gov</UIOfficeEmailAddress>
      <StateEmployerAccountNbr>1234567890</StateEmployerAccountNbr>
      <FEIN>123456789</FEIN>
      <EmployerName>ACME</EmployerName>
      <SSN>311111334</SSN>
      <ClaimantLastName>Lastname</ClaimantLastName>
      <ClaimantFirstName>Firstname</ClaimantFirstName>
      <ClaimantMiddleInitial>M</ClaimantMiddleInitial>
      <ClaimantSuffix>JR</ClaimantSuffix>
      <NumberofWeeksRequested>5</NumberofWeeksRequested>
      <EarningsVerificationWeekBeginDate>2010-08-01</EarningsVerificationWeekBeginDate>
      <EarningsVerificationWeekEndDate>2010-09-04</EarningsVerificationWeekEndDate>
      <EarningsVerificationComments>This is a comment field for this Earnings Verification Request</EarningsVerificationComments>
      <RequestDate>2010-10-14</RequestDate>
      <EarningsStatusCode>3</EarningsStatusCode>
      <TipsStatusCode>1</TipsStatusCode>
      <CommissionStatusCode>1</CommissionStatusCode>
      <BonusStatusCode>1</BonusStatusCode>
      <VacationStatusCode>1</VacationStatusCode>
      <SickLeaveStatusCode>1</SickLeaveStatusCode>
      <HolidayStatusCode>3</HolidayStatusCode>
      <SeveranceStatusCode>3</SeveranceStatusCode>
      <WagesInLieuStatusCode>4</WagesInLieuStatusCode>
      <EarningsVerificationResponseCommentIndicator>1</EarningsVerificationResponseCommentIndicator>
      <ResponseDueDate>2010-10-28</ResponseDueDate>
      <EarningsVerificationSourceCode>9</EarningsVerificationSourceCode>
   </StateEarningsVerificationRequest>
</StateEarningsVerificationRequestCollection>
```

3.4 Sample State Earnings Verification Response

```
<?xml version="1.0"?>
<EmployerTPAEarningsVerificationResponseCollection
xsi:schemaLocation="https:// REDACTED /schemas
EarningsVerificationResponse.xsd" xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <EmployerTPAEarningsVerificationResponse>
      <!-- Backfilled -->
      <StateEarningsVerificationRequestRecordGUID>AAA530000000000000000000000
00003</StateEarningsVerificationRequestRecordGUID>
      <!-- Backfilled -->
      <BrokerRecordTransactionNumber>5447</BrokerRecordTransactionNumber>
      <!-- Backfilled -->
      <RequestingStateAbbreviation>ST</RequestingStateAbbreviation>
      <!-- Backfilled -->
      <UIOfficeName>Office Name</UIOfficeName>
      <!-- Backfilled -->
      <StateEmployerAccountNbr>1234567890</StateEmployerAccountNbr>
      <!-- Backfilled -->
      <FEIN>123456789</FEIN>
      <CorrectedFEIN>987654321</CorrectedFEIN>
      <!-- Backfilled -->
      <EmployerName>ACME</EmployerName>
      <CorrectedEmployerName>Fly By Night</CorrectedEmployerName>
      <!-- Backfilled -->
      <SSN>311111334</SSN>
      <ClaimantNameWorkedAsForEmployer>John Q
Public</ClaimantNameWorkedAsForEmployer>
      <!-- Backfilled -->
      <NumberofWeeksRequested>5</NumberofWeeksRequested>
      <!-- Backfilled -->
      <EarningsVerificationWeekBeginDate>2010-08-
01</EarningsVerificationWeekBeginDate>
      <!-- Backfilled -->
      <EarningsVerificationWeekEndDate>2010-09-
04</EarningsVerificationWeekEndDate>
      <!-- 1 - Claimaint works, 20 - Never Employed Here, 21 - TPA does not
represent Employer -->
      <ClaimantEmployerWorkRelationshipCode>1</ClaimantEmployerWorkRelationsh
ipCode>
      <!-- 1 Yes, has earnings, 2 - did not have earnings (100% Sales), 9 -
No Work -->
      <EmployerEarningsCode>1</EmployerEarningsCode>
      <FirstDayWorkedinPeriod>2010-08-01</FirstDayWorkedinPeriod>
      <!-- 1 for Yes, 2 for No -->
      <StillWorkingCode>2</StillWorkingCode>
      <LastDayWorked>2010-09-04</LastDayWorked>
      <!-- 1 - Layoff, 2 - Fired, 3 - Vol Quit, 4 - Other -->
      <EmployerSepReasonCode>1</EmployerSepReasonCode>
      <!-- When Request = 1 or (2 with Work/Relationship = 20/21 or Earnings
Code = 9) -->
      <EarningsVerificationResponseComment>This employee was let go during
the time period</EarningsVerificationResponseComment>
      <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-01</WeekBeginDate>
            <WeekEndDate>2010-08-07</WeekEndDate>
            <HoursWorked>15:00</HoursWorked>
            <!-- See Request values for required or not for all below -->
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
            <EarningsPaidDate>2010-08-21</EarningsPaidDate>
            <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
            <HolidayPaidDate>2010-08-07</HolidayPaidDate>
            <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
            <SeverancePaidDate>2010-08-07</SeverancePaidDate>
           <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
            <WagesInLieuPaidDate>2010-08-07</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-08</WeekBeginDate>
            <WeekEndDate>2010-08-14</WeekEndDate>
            <HoursWorked>15:00</HoursWorked>
            <!-- See Request values for required or not for all below -->
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
            <EarningsPaidDate>2010-08-21</EarningsPaidDate>
            <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
            <HolidayPaidDate>2010-08-14</HolidayPaidDate>
            <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
            <SeverancePaidDate>2010-08-14</SeverancePaidDate>
           <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
            <WagesInLieuPaidDate>2010-08-14</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-15</WeekBeginDate>
            <WeekEndDate>2010-08-21</WeekEndDate>
            <HoursWorked>15:00</HoursWorked>
            <!-- See Request values for required or not for all below -->
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
            <EarningsPaidDate>2010-08-21</EarningsPaidDate>
            <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
            <HolidayPaidDate>2010-08-21</HolidayPaidDate>
            <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
            <SeverancePaidDate>2010-08-21</SeverancePaidDate>
           <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
            <WagesInLieuPaidDate>2010-08-21</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-22</WeekBeginDate>
            <WeekEndDate>2010-08-28</WeekEndDate>
            <HoursWorked>101:00</HoursWorked>
            <!-- See Request values for required or not for all below -->
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
            <EarningsPaidDate>2010-08-28</EarningsPaidDate>
            <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
            <HolidayPaidDate>2010-08-28</HolidayPaidDate>
            <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
            <SeverancePaidDate>2010-08-28</SeverancePaidDate>
           <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
            <WagesInLieuPaidDate>2010-08-28</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-29</WeekBeginDate>
            <WeekEndDate>2010-09-04</WeekEndDate>
            <HoursWorked>5:00</HoursWorked>
            <!-- See Request values for required or not for all below -->
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
            <EarningsPaidDate>2010-09-04</EarningsPaidDate>
            <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
            <HolidayPaidDate>2010-09-04</HolidayPaidDate>
            <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
```

```
            <SeverancePaidDate>2010-09-04</SeverancePaidDate>
           <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
            <WagesInLieuPaidDate>2010-09-04</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <!-- E - Employer, T - TPA -->
      <PreparerTypeCode>T</PreparerTypeCode>
      <PreparerCompanyName>ABC TPA</PreparerCompanyName>
      <PreparerTelephoneNumberPlusExt>5555555556</PreparerTelephoneNumberPlus
Ext>
      <PreparerContactName>Mrs Sue Herman</PreparerContactName>
      <PreparerTitle>Claims Administrator</PreparerTitle>
      <PreparerFaxNbr>5555555557</PreparerFaxNbr>
      <PreparerEmailAddress>sue.herman@abctpa.com</PreparerEmailAddress>
      <!-- Backfilled -->
      <EarningsVerificationSourceCode>9</EarningsVerificationSourceCode>
      </EmployerTPAEarningsVerificationResponse>
</EmployerTPAEarningsVerificationResponseCollection>
```

3.5 XSD

On the client connector, as well as the **Central Broker**, the request and response files must validate against the XML schema definition.

In order to implement the XML for **SIDES** data, the Exchange Formats were translated into XML Schema Definition (XSD) files. These XSD files are used in **SIDES** to validate the state request and the employer or TPA response.

Any violation of the XSD will result in an error indicating that the request or response was not successfully processed and must be fixed by the sender and resubmitted to **SIDES**.

3.5.1 Separation Information XSD

There are three files that make up this definition for Separation Information.

Two main files make up the schema:

- SeparationRequest.xsd
- SeparationResponse.xsd

One support file contains elements that are defined in both files.

- RequestResponseTypeElements.xsd

There is one XSD file, combined.xsd, which is used to include other XSD files in the system, and it does not contain any additional information. This file is required due to a problem accessing the https:// **REDACTED** /schemas namespace in multiple files within the Java libraries used in **SIDES**. The combined.xsd file is used internally by the **Central Broker** to allow XSD checks to take place on all the SOAP messages and records sent in by the connectors. The combined.xsd file may be used by connector software, but it is not necessary if the technology and libraries used in the connectors' implementation do not require it.

- Combined.xsd

3.5.1.1 XSD Files

3.5.1.1.1 SeparationRequest.xsd

See **REDACTED** for the latest copy of the SeparationRequest.xsd.

3.5.1.1.2 SeparationResponse.xsd

See **REDACTED** for the latest copy of the SeparationResponse.xsd.

3.5.1.1.3 RequestResponseTypeElements.xsd

See **REDACTED** for the latest copy of the RequestResponseTypeElements.xsd.

3.5.1.1.4 combined.xsd

See **REDACTED** for the latest copy of the combined.xsd.

3.5.2 Earnings Verification XSD

There are three files that make up this definition for Earnings Verification.

Two main files make up the schema:

- EarningsVerificationRequest.xsd

- EarningsVerificationResponse.xsd

One support file contains elements that are defined in both files.

- EarningsVerificationTypeElements.xsd

The RequestResponseTypeElements.xsd mentioned above in Separation Information is reused..

There is one XSD file, combined.xsd, which is used to include other XSD files in the system, and it does not contain any additional information. This file is required due to a problem accessing the https:// **REDACTED** /schemas namespace in multiple files within the Java libraries used in **SIDES**. The combined.xsd file is used internally by the **Central Broker** to allow XSD checks to take place on all the SOAP messages and records sent in by the connectors. The **REDACTED** file may be used by connector software, but it is not necessary if the technology and libraries used in the connectors' implementation do not require it.

3.5.2.1 XSD Files

3.5.2.1.1 EarningsVerificationRequest.xsd

See **REDACTED** for the latest copy of the EarningsVerificationRequest.xsd.

3.5.2.1.2 EarningsVerificationResponse.xsd

See **REDACTED** for the latest copy of the EarningsVerificationResponse.xsd.

3.5.2.1.3 EarningsVerificationTypeElements.xsd

See **REDACTED** for the latest copy of the EarningsVerificationTypeElements.xsd.

3.5.2.1.4 combined.xsd

See **REDACTED** for the latest copy of the combined.xsd.

3.5.3 Null/Empty Values

In many instances, the Exchange Format and the XSD indicate that an element can be null or is not required. There are two main ways to represent null values in XML (strings being a special case):

- One is to include xsi:nil=”true” if the element in question is supposed to be null
- The other is to not include the element

For **SIDES**, the way to indicate null values is to not include the element. Therefore, any element that does not have a value must **not** appear in the XML file sent to the **Central Broker**.

The ClaimantFirstName and ClaimantLastName data elements in the Separation Request are required. Yet, there may not be a value that can be placed in them (in the case where the claimant does not have either a first or last name); if this occurs, a space must be sent in as the element value.

3.5.4 Dates

The XSD defines all of the Date data types as *xs:date*. This data type allows the definition of a year, month, and a day to define the particular date. The Exchange Format date is restricted to just the year, month and day (10 characters total). It is important to keep the *xs:date* field to just the year, month, and day.

There are two exceptions to this rule where the full date/time is used. This is in the case of the Broker Effective Date field and the DateStartedReceivingTransmission/ DateFinishedReceivingTransmission fields in the message acknowledgements. The Broker Effective Date indicates when a record was received in the **Central Broker** and requires the use of the date and time fields in *xs:dateTime* to record the exact time of record receptions and transmissions by the Broker. The DateStartedReceivingTransmission/ DateFinishedReceivingTransmission are part of the acknowledgements and give the receiver the date and time box around the transmission. The time zones for each of these elements will always be in Greenwich Mean Time or an offset thereof.

3.5.5 MTOM

Message Transmission Optimization Mechanism, or MTOM, is a mechanism for transmitting large binary attachments with SOAP messages as raw bytes, allowing for smaller messages. Binary content often has to be re-encoded to be sent as text data with SOAP messages. MTOM allows more efficient sending of binary data in a SOAP request or response. MTOM provides a way of efficiently transmitting binary data such as images, PDF files, and MS Word documents, between connectors.

The basis of MTOM used is the data type base64Binary (http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#base64Binary).

This is defined in the Separation Request and Separation Response XSD as part of the attachment occurrence:

```
<xs:element name="AttachmentData" type="xs:base64Binary" />
```

Connectors must use the base64Binary data type for their request/response attachments.

3.5.6 Backfilled Data

There are some backfilled data required in the response. This data shall come out of the request record in the exact form it is received. **Central Broker** business rule checks determine if the data matches between the request and response on these fields. If there is any difference, the **Central Broker** rejects the record.

The fields that must be backfilled are given in Table 2 and 3.

Table 2 – Separation Information Backfilled Data

Element Name	Separation Information Exchange Format Sequence Number	Notes
StateRequestRecordGUID	B-59	

Element Name	Separation Information Exchange Format Sequence Number	Notes
BrokerRecordTransactionNumber	B-60	
SSN	B-1	
ClaimEffectiveDate	B-2	
ClaimNumber	B-3	If this was not included in the separation request, then it must not be included in the separation response.
StateEmployerAccountNbr	B-4	

Table 3 – Earnings Verification Backfilled Data

Element Name	Earnings Verification Exchange Format Sequence Number	Notes
StateEarningsVerificationRequestRecordGUID	ER-1	
BrokerRecordTransactionNumber	ER-2	
RequestingStateAbbreviation	ER-3	
UIOfficeName	ER-4	
StateEmployerAccountNbr	ER-5	
FEIN	ER-6	
EmployerName	ER-8	
SSN	ER-10	
NumberofWeeksRequested	ER-12	

Element Name	Earnings Verification Exchange Format Sequence Number	Notes
EarningsVerificationWeekBeginDate	ER-13	
EarningsVerificationWeekEndDate	ER-14	
EarningsVerificationSourceCode	ER-32	

3.6 File Size

States may include multiple requests in a single XML file per employer or TPA to which they want to send requests. This file may be up to 8MB in size, including all encoded attachments, prior to encryption. If states have more than 8MB of data for that employer or TPA, they must create multiple files.

Similarly, multiple employer/TPA responses to a state may be packaged into a single file within the 8MB limit. If employers or TPAs have more than 8MB of data for a state (including encoded attachments but prior to encryption), they must also create multiple files.

3.7 Business Rules

The Business Rules column in the Exchange Formats contains additional business rule validation logic ("edits") that cannot be defined in an XSD. To keep the data in **SIDES** consistent, the **Central Broker** uses Java code to check every request and response record that is sent to it. The **Central Broker** verifies that all of the business rules are followed. If it detects any of these rules have been violated, it passes back an error code and message to the calling client that indicates the rules that were violated.

An example of a Java implemented business rule (rather than a validation that can be implemented within the XSD) is given below for the Separation Information exchange. This example logs error 219 (see Part B – Connector Requirements) if the associated business rule is violated.

```
if ((sep.getWorkingAllAvailableHoursInd() == null)

        && sep.getEmployerSepReasonCode()!= null

        && (sep.getEmployerSepReasonCode().intValue() == 11))

{

    // Error
```

```
        errorList.add(new BRValidationError(219));

        logger.debug("Found error (219) in
        WorkingAllAvailableHoursInd - it is null when
        EmployerSepReasonCode equals 11");

    }
```

In order to create a connector, the software must implement and execute all of the validations and business rules specified in the Exchange Formats prior to sending the request or response to the Broker.

4 B – BUILD THE CONNECTOR: PREPARING THE MESSAGE

Prior to transmittal of the request or response data to the **Central Broker**, the connector must extract the data from its backend system and generate an XML file. Once the XML file has been created, the file must be packaged and delivered to the **Central Broker**. This section describes the messaging framework that a connector can use to deliver messages to and receive messages from the **Central Broker**.

4.1 Messaging Overview – Post, Pull, and Push

The communication between the **Central Broker** and the state, employer, and TPA connectors is accomplished through SOAP over HTTPS using an HTTP request/response pattern.

There are three operations supported by the **Central Broker** – "Post," "Pull," and "Push."

4.1.1 Post

In the Post operation, the connecting client instigates communication with the Broker. The connectors "Post" their request and response files to the **Central Broker** (HTTP request) and receive an acknowledgement in return (HTTP response).

Post Transaction:

1. Connector posts its request file (if a state), or response file (if an employer or TPA) to the Broker in an HTTP request

2. Connector receives receipt of file in an HTTP response from the Broker

4.1.2 Pull

In the Pull operation, the connector asks for any available records from the **Central Broker** (HTTP request), receives the waiting records (HTTP response) and sends back an acknowledgement in return (HTTP request). Because there are three communications that take place with this action, the process is broken up into two distinct HTTP request/response transactions:

Pull Transaction (1):

1. Connector asks for its files (responses if a state, requests if an employer or TPA) in an HTTP request to the Broker

2. Connector receives its files in an HTTP response from the Broker

Pull Transaction (2):

3. Connector acknowledges its receipt of file in an HTTP request to the Broker

> ***Note*:** There is no HTTP response to the receipt HTTP transaction (number 3 above). The Broker simply records the reception of the receipt HTTP request or logs an error. (There is no "receipt to the receipt.")

4.1.3 Push

In the "Push" transaction, the Broker instigates communication to the employer or TPA client. Immediately following each state "Post," the Broker processes the incoming request file and then sends it to the employer or TPA connector Web service instantaneously.

The "Push" transaction requires that the employer or TPA have a listening Web service and the Broker be configured to operate in Push mode for that employer or TPA.

Push Transaction:

1. Broker pushes request file to employer or TPA in an HTTP request to the employer or TPA connector, which has been implemented and configured to listen for Broker pushes

2. Broker receives receipt of a pushed request file in a HTTP response from the employer/TPA

The "Push" transaction only occurs from Broker to employer or TPA for request files from the states. There is no equivalent Broker-to-state "Push" for the "automatic" delivery of employer or TPA response files to the states at this time. State connectors must use "Pull" to retrieve their response files.

Employers and TPAs may choose to implement Pull or Push for their response files. This is a free choice for each employer and TPA depending upon the process they wish to implement for their backend systems.

The employer and TPA choice for Pull or Push transaction has no effect on the state clients. It affects only the configuration of the Broker and whether a particular employer or TPA has a client that implements Pull or Push mode. The state posts its requests to the Broker in the same way, regardless of whether the employer or TPA receives them from the Broker via Pull or Push.

4.1.4 SOAP

SOAP is a protocol for exchange of information in a decentralized, distributed environment. It is an XML-based protocol that consists of three parts: (1) an envelope that defines a framework for describing what is in a message and how to process it, (2) a set of encoding rules for expressing instances of application-defined data types, and (3) a convention for representing remote procedure calls and responses.

SOAP messages are fundamentally one-way transmissions from a sender to a receiver. But SOAP messages are often combined to implement patterns such as the request/response pattern, where it provides for SOAP response messages to be delivered as HTTP responses, using the same connection as the inbound request. This is the pattern used to accomplish the file transfer/acknowledgement scheme, as overviewed in Sections 4.1.1, 4.1.2, and 4.1.3.

4.2 Messaging Concepts

There are a few concepts that need to be discussed before getting further into the messaging process for SIDES.

4.2.1 Unique ID

Each state, employer, and TPA will be assigned a Unique ID for both the SIDES production environment and the SIDES test environment. Please contact the SIDES Business Manager to obtain your Unique IDs for SIDES. A complete list of Unique IDs is maintained on the SIDES Website (http://SIDES.itsc.org). The Unique IDs of all the current participants are as follows:

4.2.1.1 State Unique IDs

Table 4 lists the participating states and their Unique IDs:

Table 4 - Unique IDs of Current Participating States

State	Unique ID
REDACTED	

4.2.1.2 Employer/TPA Unique IDs

Table 5 lists the participating employers and TPAs and their Unique IDs. The Employer/TPA Unique ID is a 'BR' followed by a unique nine digit number. The **Broker Administrator** assigns the nine digit number.

Table 5 - Unique IDs of Current Participating Employer/TPAs

Employer/TPA	Unique ID
REDACTED	

4.2.2 File and Record GUIDs

SIDES uses a Globally Unique Identifier (GUID). A GUID is a special type of identifier used in software applications to provide a unique reference number. The value is represented as a 32 character hexadecimal character string, such as {21EC2020-3AEA-1069-A2DD-08002B30309D}. The primary purpose of the GUID is to have a totally unique number. Ideally, a GUID will never be generated twice by any computer or group of computers in existence. The total number of unique keys (2^{128} or 3.4×10^{38}) is so large that the probability of the same number being generated twice is extremely small. The SIDES team suggests GUIDs be generated using utilities or function calls available within your development framework.

GUIDs are created by connectors for each "Post" transaction to the Broker and are used on both the XML file level (File GUIDs) and the XML individual record level (Record GUIDs).

The File GUIDs are used on an entire XML file to uniquely identify that file.

NOTE: States must not reuse a GUID unless it is was used more than 10 years ago. GUIDs may only be reused for a resend of a request that was rejected by the **Central Broker**.

The Record GUIDs are used within the XML on each record to uniquely identify that record. Record GUIDs should be unique with the domain of the connector and not just that file.

There are many GUID creation methods depending on the language and technology in use. Many of them will create the GUID with a '-' in a few places such that it actually comes out to be 36 characters long. Developers of connectors must be aware that as the GUID for SIDES is defined as a 32-character string, the '-' must be stripped before its use.

NOTE: Because of the small chance that a GUID will be repeated by different states, employer/TPAs must ensure that both the State Request Record GUID **AND** the State abbreviation (or something similar) be used when determining if a record is a duplicate.

4.2.3 SOAP Transaction Number

The **Central Broker** uses a SOAP transaction number (StateSOAPTransactionNumber and EmployerTPASOAPTransactionNumber) as a unique identifier for a file as part of a "Pull" transaction. These can be thought of like a FedEx or UPS tracking number. These numbers can be used in calls to re-Pull a particular file in case of loss.

4.2.4 Broker Record Transaction Number

The BrokerRecordTransactionNumber is given to a single request record on entry into the system and is generated by the Broker. This number uniquely identifies a request, even if multiple copies of the same record are passed through the system (in this case, each record gets its own BrokerRecordTransactionNumber). The BrokerRecordTransactionNumber must be used on the response in order to connect the response with a particular request and, therefore, must be consumed by employer and TPA connectors.

4.2.5 Message Codes

Each time an acknowledgement is sent, a message code is sent with it to indicate the status of the transaction. These are the transmission-level message codes that go in the SOAP header as discussed below in Section 4.3 - SOAP Custom Headers.

4.2.5.1 Post-Acknowledgement Message Codes

Table 6 – Post-Acknowledgement Message Codes

Code	Message	Notes
1	File Success	Successful Transmission; no rejects
2	File Failure	File size too large; no records in file; all records failed
3	File Success with Rejected Records	Rejected records included

4.2.5.2 Pull-Response Message Codes

Table 7 – Pull-Response Message Codes

Code	Message	Notes
1	File Contained in Payload	The file is contained in the payload of the SOAP Message
2	End Of Files	There are no files available to download

4.2.5.3 Pull Acknowledgement Codes

Table 8 - Pull Acknowledgement Codes

Code	Message	Notes
1	File Success	Successful transmission
2	File Failure	Did not receive file; any file

		problems

4.3 SOAP Custom Headers

This section discusses the contents of the custom SOAP headers of the various transactions that occur in the Post, Pull, and Push processes.

The SOAP headers are one part of a SOAP message. The SOAP messages used contain additional custom information in the SOAP headers. This information utilizes the messaging concepts discussed in the previous Section 4.2 Messaging Concepts. Each type of transaction has its own requirements and elements as discussed below.

The custom SOAP header information provided by the connecting client to the Broker is for routing purposes, security purposes and, in the case of the Pull, the type of Pull required.

The custom SOAP header information provided by the Broker to the client is for security purposes and, in the case of the Pull, for re-Pulling purposes.

The SOAP headers provided for each type of transaction are listed below.

- State Post (Section 4.3.1)
    - State Post to Broker (for routing to specified employer or TPA)
    - State Post to Broker (for routing to the **SIDES** Employer Web site)
    - Broker Acknowledgement to state Post
- State Pull from Broker – Regular Pull (Section 4.3.2.1)
    - State Request to Broker (Regular Pull)
    - Broker Response to Request (Regular Pull)
    - State Acknowledgement to Broker (Regular Pull)
- State Pull from Broker – Re-Pull by StateSOAPTransactionNumber (Section 4.3.2.2)
    - State Request to Broker (Re-Pull by StateSOAPTransactionNumber)
    - Broker Response to Request (Re-Pull by StateSOAPTransactionNumber)
    - State Acknowledgement to Broker (Re-Pull by StateSOAPTransactionNumber)
- State Pull from Broker – Re-Pull by Date (Section 4.3.2.3)
    - State Request to Broker (Re-Pull by Date)

    - Broker Response to Request (Re-Pull by Date)
    - State Acknowledgement to Broker (Re-Pull by Date)
- Employer/TPA Post (Section 4.3.3)
    - Employer/TPA Post to Broker
    - Broker Acknowledgement to Employer/TPA Post
- Employer/TPA Pull from Broker – Regular Pull (Section 4.3.4.1)
    - Employer/TPA Request to Broker (Regular Pull)
    - Broker Response to Request (Regular Pull)
    - Employer/TPA Acknowledgement to Broker (Regular Pull)
- Employer/TPA Pull from Broker – Re-Pull by EmployerTPASOAPTransactionNumber (Section 4.3.4.2)
    - Employer/TPA Request to Broker (Re-Pull by EmployerTPASOAPTransactionNumber)
    - Broker Response to Request (Re-Pull by EmployerTPASOAPTransactionNumber)
    - Employer/TPA Acknowledgement to Broker (Re-Pull by EmployerTPASOAPTransactionNumber)
- Employer/TPA Pull from Broker – Re-Pull by Date (Section 4.3.4.3)
    - Employer/TPA Request to Broker (Re-Pull by Date)
    - Broker Response to Request (Re-Pull by Date)
    - Employer or TPA Acknowledgement to Broker (Re-Pull by Date)

4.3.1 State Post

The following sections contain the custom header elements for a state Post to the Broker. Note that the tables below that describe messages going to the Broker have a column that indicates which fields are required. Tables that describe messages being returned from the Broker do not have this column, as there is no responsibility on the client connector to populate these fields. The client connector must handle whatever is returned by the Broker according to the header specification.

4.3.1.1 State Post to Central Broker

Table 9 - State Post to Broker

Header Element	Required	Definition	Example
To	Y	The Unique ID of the employer or TPA to which the message is intended Will always be 'BR' followed by nine digits	BR000000003
From	Y	The Unique ID of the state where the message originated	UT
StateRequestFileGUID	Y	The state-generated GUID applied to this message that can uniquely identify this file Size is 32 hexadecimal digits	A42A1FBDAC9549AC7D8D3F45E4040319

4.3.1.1.1 SOAP Example - State Post to Central Broker:

```
<To xmlns="https://REDACTED/schemas">BR000000003</To>
<From xmlns="https://REDACTED/schemas">UT</From>
<StateRequestFileGUID
xmlns="https://REDACTED/schemas">
A42A1FBDAC9549AC7D8D3F45E4040319</StateRequestFileGUID>
```

4.3.1.2 State Post to Central Broker – SIDES Employer Website

Table 10 - State Post to Broker - SIDES Employer Website

Header Element	Required	Definition	Example
To	Y	The FEIN of the employer or TPA to which the message is intended Size is nine numeric digits	123456789
From	Y	The Unique ID of the state where the message originated	NJ

Header Element	Required	Definition	Example
StateRequestFileGUID	Y	The state-generated GUID applied to this message that can uniquely identify this file Size is 32 hexadecimal digits	A42A1FBDAC9549AC7D8D3F45E4040319
Separation Information Only SEIN	Y	The SEIN of the employer or TPA to which the message is intended. For those states that do not use the SEIN, this must equal the FEIN Size is up to 20 digits	123456789
PIN	Y	The PIN to which the state wants to assign this request for this employer or TPA Size is up to 20 characters	435222169876

4.3.1.2.1 SOAP Example - State Post to Central Broker – SIDES Employer Website:

```
<To xmlns="https://REDACTED/schemas">123456789</To>
<From xmlns="https://REDACTED/schemas">NJ</From>
<StateRequestFileGUID
xmlns="https://REDACTED /schemas">
A42A1FBDAC9549AC7D8D3F45E4040319</StateRequestFileGUID>
<SEIN xmlns="https://REDACTED/schemas">123456789</From>
<PIN xmlns="https://REDACTED/schemas">435222169876</From>
```

4.3.1.3 Central Broker Acknowledgement to State Post

Table 11 - Broker Acknowledgement to State Post

Header Element	Definition	Example
To	The Unique ID of the state that sent the Post	UT
From	Will always be “Broker”	Broker
StateRequestFileGUID	The state-generated GUID applied to the message that uniquely identifies the file sent in to the Broker. This is for verification purposes	A42A1FBDAC9549AC7D8D3F45E4040319

Header Element	Definition	Example
	Size is 32 hexadecimal digits	
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.1.3.1 SOAP Example - Central Broker Acknowledgement to State Post:

```
<To xmlns="https:// REDACTED/schemas">UT</To>
<From xmlns="https:// REDACTED/schemas">Broker</From>
<StateRequestFileGUID xmlns="https://REDACTED
/schemas">A42A1FBDAC9549AC7D8D3F45E4040319</StateRequestFileGUID>
<MessageCode xmlns="https:// REDACTED/schemas">1</MessageCode>
```

4.3.2 State Pull

Because of the nature of the HTTP request-response pattern, there is one request and one response for each HTTP request-response transaction. Because three messages are sent between the connector and the Broker on a Pull, there will be two request-response patterns needed to accomplish the full Pull operation. (See Section 4.1.2 for an overview of this issue.)

4.3.2.1 State Pull from Central Broker – Regular Pull

The following sections contain the custom header elements for a regular state Pull from the Broker.

4.3.2.1.1 State Request to Central Broker (Regular Pull)

Table 12 - State Request to Broker (Regular Pull)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the state where this message originated	UT
PullCollection	Y	Signifies one of three Pull transactions desired by the state 1 - indicates a regular Pull	1

		Size is one digit	

4.3.2.1.1.1 SOAP Example - State Request to Central Broker (Regular Pull):

```
<To xmlns="https://REDACTED/schemas">Broker</To>
<From xmlns="https://REDACTED/schemas">UT</From>
<PullCollection xmlns="https://REDACTED/schemas">1</PullCollection>
```

4.3.2.1.2 Central Broker Response to Request (Regular Pull)

Table 13 - Broker Response to Request (Regular Pull)

Header Element	Definition	Example
To	The Unique ID of the state that requested the Pull	UT
From	The Unique ID of the employer or TPA from which these response records originated	BR000000001
StateSOAPTransactionNumber	The unique number assigned to this file by the Broker	3565
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.2.1.2.1 SOAP Example – Central Broker Response to Request (Regular Pull):

```
<To xmlns="https://REDACTED/schemas">UT</To>
<From xmlns="https:// REDACTED/schemas">BR000000001</From>
<StateSOAPTransactionNumber xmlns="https://
REDACTED/schemas">3565</StateSOAPTransactionNumber>
<MessageCode xmlns="https://REDACTED/schemas">1</MessageCode>
```

4.3.2.1.3 State Acknowledgment to Central Broker (Regular Pull)

Table 14 - State Acknowledgement to Broker (Regular Pull)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the state from which this message originated	UT
StateSOAPTransactionNumber	Y	The StateSOAPTransactionNumber that was returned in the response for the regular pull	3565
MessageCode	Y	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.2.1.3.1 SOAP Example – State Acknowledgment to Central Broker (Regular Pull):

```
<To xmlns="https:// REDACTED/schemas">Broker</To>
<From xmlns="https:// REDACTED/schemas">UT</From>
<StateSOAPTransactionNumber xmlns="https://
REDACTED/schemas">3565</StateSOAPTransactionNumber>
<MessageCode xmlns="https://REDACTED/schemas">1</MessageCode>
```

4.3.2.2 State Pull from Central Broker – Re-Pull by StateSOAPTransactionNumber

The following sections contain the custom header elements for a State Re-Pull by StateSOAPTransactionNumber from the Broker.

4.3.2.2.1 State Request to Central Broker (Re-Pull by StateSOAPTransactionNumber)

Table 15 - State Request to Broker (Re-Pull by StateSOAPTransactionNumber)

Header Element	Required	Definition	Example
To	Y	Broker	Broker

Header Element	Required	Definition	Example
From	Y	The Unique ID of the state from which this message originated	UT
PullCollection	Y	Signifies one of three Pull transactions desired by the state 2 - Indicates a re-Pull by StateSOAPTransactionNumber Size is one digit	2
StateSOAPTransactionNumber	Y	The StateSOAPTransactionNumber that was returned in the response for the regular pull on a previous Pull request. This specifies the file the State wants to re-Pull	3565

4.3.2.2.1.1 SOAP Example - State Request to Central Broker (Re-Pull by StateSOAPTransactionNumber):

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas">UT</From>
<PullCollection xmlns="https:// REDACTED /schemas">2</PullCollection>
<StateSOAPTransactionNumber xmlns="https:// REDACTED
/schemas">3565</StateSOAPTransactionNumber>
```

4.3.2.2.2 Central Broker Response to Request (Re-Pull by StateSOAPTransactionNumber)

Table 16 - Broker Response to Request (Re-Pull by StateSOAPTransactionNumber)

Header Element	Definition	Example
To	The Unique ID of the state from which this message originated	UT
From	The Unique ID of the employer or TPA from which these response records originated	BR000000001
StateSOAPTransactionNumber	The StateSOAPTransactionNumber that was requested by the state	3565

Header Element	Definition	Example
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.2.2.2.1 SOAP Example - Central Broker Response to Request (Re-Pull by StateSOAPTransactionNumber):

```
<To xmlns="https:// REDACTED /schemas">UT</To>
<From xmlns="https:// REDACTED /schemas">BR000000001</From>
<StateSOAPTransactionNumber xmlns="https:// REDACTED
/schemas">3565</StateSOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.3.2.2.3 State Acknowledgment to Central Broker (Re-Pull by StateSOAPTransactionNumber)

Table 17 -State Acknowledgment to Broker (Re-Pull by StateSOAPTransactionNumber)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the state from which this message originated	UT
StateSOAPTransactionNumber	Y	The StateSOAPTransactionNumber that was returned in the response for the regular pull	3565
MessageCode	Y	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes.	1

Header Element	Required	Definition	Example
		Size is one digit	

4.3.2.2.3.1 SOAP Example - State Acknowledgment to Central Broker (Re-Pull by StateSOAPTransactionNumber):

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas">UT</From>
<StateSOAPTransactionNumber xmlns="https:// REDACTED
/schemas">3565</StateSOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.3.2.3 State Pull from Central Broker – Re-Pull by Date

The following sections contain the custom header elements for a State Re-Pull by Date from the Broker.

4.3.2.3.1 State Request to Central Broker (Re-Pull by Date)

Table 18 - State Request to Broker (Re-Pull by Date)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the state from which this message originated	UT
PullCollection	Y	Signifies one of three Pull transactions desired by the state 3 - Indicates a re-Pull by Date Size is one digit	3
StateSOAPTransactionNumber	Y	This must not be included for the first call by Date. On subsequent calls, this must be filled in with the NextStateSOAPTransactionNumber returned on the previous call in order to collect all records on the date specified	

4.3.2.3.1.1 SOAP Example - State Request to Central Broker (Re-Pull by Date):

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas">UT</From>
<PullCollection xmlns="https:// REDACTED /schemas">3</PullCollection>
```

4.3.2.3.2 Central Broker Response to Request (Re-Pull by Date)

Table 19 - Broker Response to Request (Re-Pull by Date)

Header Element	Definition	Example
To	Broker	Broker
From	The Unique ID of the state from which this message originated	UT
StateSOAPTransactionNumber	The first StateSOAPTransactionNumber that was returned in the response for the regular pull on that date	3565
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1
NextStateSoapTransactionNumber	The next StateSOAPTransactionNumber that was returned in the response for the regular pull on that date. This header will not be included in the response when the last file in the data range is being returned, indicating there are no more files to be sent.	3566

4.3.2.3.2.1 SOAP Example - Central Broker Response to Request (Re-Pull by Date):

```
<To xmlns="https:// REDACTED /schemas">UT</To>
<From xmlns="https:// REDACTED /schemas">BR000000001</From>
<StateSOAPTransactionNumber xmlns="https:// REDACTED
/schemas">3565</StateSOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
<NextStateSOAPTransactionNumber xmlns="https:// REDACTED
/schemas">3566</NextStateSOAPTransactionNumber>
```

4.3.2.3.3 State Acknowledgment to Central Broker (Re-Pull by Date)

Table 20 - State Acknowledgement to Broker (Re-Pull by Date)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the state from which this message originated	UT
StateSOAPTransactionNumber	Y	The StateSOAPTransactionNumber that was returned in the response for the regular pull	3565
MessageCode	Y	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit.	1

4.3.2.3.3.1 SOAP Example - State Acknowledgment to Central Broker (Re-Pull by Date)

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas">UT</From>
<StateSOAPTransactionNumber xmlns="https:// REDACTED
/schemas">3565</StateSOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.3.3 Employer/TPA Post

The following sections contain the custom header elements for an employer or TPA Post to the Broker.

4.3.3.1 Employer/TPA Post to Central Broker

Table 21 - Employer/TPA Post to Broker

Header Element	Required	Definition	Example
To	Y	The Unique ID of the state to which this	UT

Header Element	Required	Definition	Example
		message is intended	
From	Y	The Unique ID of the employer or PA from which the message originated Will always be ‘BR’ followed by nine digits	BR000000003
EmployerTPAResponseFileGUID	Y	The employer or TPA-generated GUID applied to this message that can uniquely identify this file Size is 32 hexadecimal digits	A42A1FBDAC9549AC7D8D3F45E4040319

4.3.3.1.1 SOAP Example - Employer/TPA Post to Central Broker:

```
<To xmlns="https:// REDACTED /schemas"> UT </To>
<From xmlns="https:// REDACTED /schemas"> BR000000003</From>
<EmployerTPAResponseFileGUID
xmlns="https:// REDACTED
/schemas">A42A1FBDAC9549AC7D8D3F45E4040319</EmployerTPAResponseFileGUID>
```

4.3.3.2 Central Broker Acknowledgement to Employer/TPA Post

Table 22 - Broker Acknowledgement to Employer/TPA Post

Header Element	Definition	Example
To	The Unique ID of the employer or TPA that sent the Post	BR000000003
From	Will always be “Broker.”	Broker
EmployerTPAResponseFileGUID	The employer or TPA-generated GUID applied to the message that uniquely identifies the file sent to the Broker. This is for verification purposes. Size is 32 hexadecimal digits	A42A1FBDAC9549AC7D8D3F45E4040319
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.3.2.1 SOAP Example - Central Broker Acknowledgement to Employer/TPA Post:

```
<To xmlns="https:// REDACTED /schemas">BR000000003</To>
<From xmlns="https:// REDACTED /schemas">Broker</From>
<EmployerTPAResponseFileGUID xmlns="https:// REDACTED
/schemas">A42A1FBDAC9549AC7D8D3F45E4040319</EmployerTPAResponseFileGUID>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.3.4 Employer/TPA Pull

Because of the nature of the HTTP request-response pattern, there is one request and one response for each HTTP request-response transaction. Because three messages are sent between the connector and the Broker on a Pull, there will be two request-response patterns needed to accomplish the full Pull operation. (See Section 4.1.2 for an overview of this issue.)

4.3.4.1 Employer/TPA Pull from Central Broker – Regular Pull

The following sections contain the custom header elements for a regular state Pull from the Broker.

4.3.4.1.1 Employer/TPA Request to Central Broker (Regular Pull)

Table 23 - Employer/TPA Request to Broker (Regular Pull)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the employer or TPA from which this message originated	BR000000003
PullCollection	Y	Signifies one of three Pull transactions desired by the employer or TPA 1 - Indicates a regular Pull Size is one digit	1

4.3.4.1.1.1 SOAP Example - Employer/TPA Request to Central Broker (Regular Pull):

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas"> BR000000003</From>
<PullCollection xmlns="https:// REDACTED /schemas">1</PullCollection>
```

4.3.4.1.2 Central Broker Response to Request (Regular Pull)

Table 24 - Broker Response to Request (Regular Pull)

Header Element	Definition	Example
To	The Unique ID of the employer or TPA that requested the Pull	BR000000003
From	The Unique ID of the state from which these request records originated.	UT
EmployerTPASOAPTransactionNumber	The unique number assigned to this file by the Broker	7350
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.4.1.2.1 SOAP Example – Central Broker Response to Request (Regular Pull):

```
<To xmlns="https:// REDACTED /schemas"> BR000000003</To>
<From xmlns="https:// REDACTED /schemas">UT</From>
<EmployerTPASOAPTransactionNumber xmlns="https:// REDACTED
/schemas">7350</EmployerTPASOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.3.4.1.3 Employer/TPA Acknowledgment to Central Broker (Regular Pull)

Table 25 - Employer/TPA Acknowledgment to Broker (Regular Pull)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the employer or TPA from which this message originated	BR000000003
EmployerTPASOAPTransactionNumber	Y	The EmployerTPASOAPTransactionNumber that was returned in the response for the regular Pull	7350

Header Element	Required	Definition	Example
MessageCode	Y	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.4.1.3.1 SOAP Example – Employer/TPA Acknowledgment to Central Broker (Regular Pull):

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas"> BR000000003</From>
<EmployerTPASOAPTransactionNumber xmlns="https:// REDACTED
/schemas">7350</EmployerTPASOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.3.4.2 Employer/TPA Pull from Central Broker – Re-Pull by EmployerTPASOAPTransactionNumber

The following sections contain the custom header elements for an employer or TPA Re-Pull by EmployerTPASOAPTransactionNumber from the Broker.

4.3.4.2.1 Employer/TPA Request to Central Broker (Re-Pull by EmployerTPASOAPTransactionNumber)

Table 26 - Employer/TPA Request to Broker (Re-Pull by EmployerTPASOAPTransactionNumber)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the employer or TPA from which this message originated	BR000000001
PullCollection	Y	Signifies one of three Pull transactions desired by the employer or TPA 2 - Indicates a re-Pull by	2

Header Element	Required	Definition	Example
		EmployerTPASOAPTransactionNumber Size is one digit	
EmployerTPASOAPTransactionNumber	Y	The EmployerTPASOAPTransactionNumber that was returned in the response for the regular Pull on a previous Pull request. This specifies the file the employer or TPA wants to re-Pull	7350

4.3.4.2.1.1 SOAP Example - Employer/TPA Request to Central Broker (Re-Pull by EmployerTPASOAPTransactionNumber):

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas"> BR000000001</From>
<PullCollection xmlns="https:// REDACTED /schemas">2</PullCollection>
<EmployerTPASOAPTransactionNumber xmlns="https:// REDACTED
/schemas">7350</EmployerTPASOAPTransactionNumber>
```

4.3.4.2.2 Central Broker Response to Request (Re-Pull by EmployerTPASOAPTransactionNumber)

Table 27 - Broker Response to Request (Re-Pull by EmployerTPASOAPTransactionNumber)

Header Element	Definition	Example
To	The Unique ID of the employer or TPA from which this message originated	BR000000001
From	The Unique ID of the employer or TPA from which these response records originated	UT
EmployerTPASOAPTransactionNumber	The EmployerTPASOAPTransactionNumber that was requested by the employer or /TPA	7350
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further	1

Header Element	Definition	Example
	information on Message Codes. Size is one digit.	

4.3.4.2.2.1 SOAP Example - Central Broker Response to Request (Re-Pull by EmployerTPASOAPTransactionNumber):

```
<To xmlns="https:// REDACTED /schemas"> BR000000001</To>
<From xmlns="https:// REDACTED /schemas">UT</From>
<EmployerTPASOAPTransactionNumber xmlns="https:// REDACTED
/schemas">7350</EmployerTPASOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.3.4.2.3 Employer/TPA Acknowledgment to Central Broker (Re-Pull by EmployerTPASOAPTransactionNumber)

Table 28 - Employer/TPA Acknowledgment to Broker (Re-Pull by EmployerTPASOAPTransactionNumber)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the employer or TPA from which this message originated.	BR000000001
EmployerTPASOAPTransactionNumber	Y	The EmployerTPASOAPTransactionNumber that was returned in the response for the regular Pull Size is 32 hexadecimal digits	7350
MessageCode	Y	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.4.2.3.1 SOAP Example - Employer/TPA Acknowledgment to Central Broker (Re-Pull by EmployerTPASOAPTransactionNumber):

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas"> BR000000001</From>
<EmployerTPASOAPTransactionNumber xmlns="https:// REDACTED
/schemas">7350</EmployerTPASOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.3.4.3 Employer/TPA Pull from Central Broker – Re-Pull by Date

The following sections contain the custom header elements for an employer or TPA Re-Pull by Date from the Broker.

4.3.4.3.1 Employer/TPA Request to Central Broker (Re-Pull by Date)

The first time this operation is called, the EmployerTPASOAPTransactionNumber is null and the dates from which the connector wants to Re-Pull are included in the SOAP payload.

When the Broker replies with the first file, the Broker will include the next EmployerTPASOAPTransactionNumber during that date range in a SOAP header attribute (NextEmployerTPASOAPTransactionNumber).

In the next call to this operation, the caller includes this EmployerTPASOAPTransactionNumber with the date range. This differentiates to the Broker the next call in the series from a brand new Re-Pull by Date request.

The last file sent back to the connector is indicated by a null value for the next EmployerTPASOAPTransactionNumber.

Table 29 - Employer/TPA Request to Broker (Re-Pull by Date)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the employer or TPA from which this message originated.	BR000000001
PullCollection	Y	Signifies one of three Pull transactions desired by the employer or TPA 3 - Indicates a re-Pull by Date Size is one digit	3

Header Element	Required	Definition	Example
EmployerTPASOAPTransactionNumber	Y	This must not be included for the first call by Date. On subsequent calls, this must be filled in with the NextEmployerTPASOAPTransactionNumber returned on the previous call in order to collect all records on the date specified	

4.3.4.3.1.1 SOAP Example - Employer/TPA Request to Central Broker (Re-Pull by Date):

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas"> BR000000001</From>
<PullCollection xmlns="https:// REDACTED /schemas">3</PullCollection>
```

4.3.4.3.2 Central Broker Response to Request (Re-Pull by Date)

Table 30 - Broker Response to Request (Re-Pull by Date)

Header Element	Definition	Example
To	Broker	Broker
From	The Unique ID of the employer or TPA from which this message originated.	BR000000001
EmployerTPASOAPTransactionNumber	The first EmployerTPASOAPTransactionNumber that was returned in the response for the regular Pull on that date	7350
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1
NextEmployerTPASOAPTransactionNumber	The next EmployerTPASOAPTransactionNumber that was returned in the response for the regular Pull on that date. This header will not be included in the response when the last file in the data range is being returned, indicating	7351

Header Element	Definition	Example
	there are no more files to be sent.	

4.3.4.3.2.1 SOAP Example - Central Broker Response to Request (Re-Pull by Date):

```
<To xmlns="https:// REDACTED /schemas"> BR000000001</To>
<From xmlns="https:// REDACTED /schemas">UT</From>
<EmployerTPASOAPTransactionNumber xmlns="https:// REDACTED
/schemas">7350</EmployerTPASOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
<NextEmployerTPASOAPTransactionNumber xmlns="https:// REDACTED
/schemas">7351</NextEmployerTPASOAPTransactionNumber>
```

4.3.4.3.3 Employer/TPA Acknowledgment to Central Broker (Re-Pull by Date)

Table 31 - Employer/TPA Acknowledgment to Broker (Re-Pull by Date)

Header Element	Required	Definition	Example
To	Y	Broker	Broker
From	Y	The Unique ID of the employer or TPA from which this message originated.	BR000000001
EmployerTPASOAPTransactionNumber	Y	The EmployerTPASOAPTransactionNumber that was returned in the response for the regular Pull	7350
MessageCode	Y	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

4.3.4.3.3.1 SOAP Example - Employer/TPA Acknowledgment to Central Broker (Re-Pull by Date)

```
<To xmlns="https:// REDACTED /schemas">Broker</To>
<From xmlns="https:// REDACTED /schemas"> BR000000001</From>
```

```
<EmployerTPASOAPTransactionNumber xmlns="https:// REDACTED
/schemas">7350</EmployerTPASOAPTransactionNumber>
<MessageCode xmlns="https:// REDACTED /schemas">1</MessageCode>
```

4.4 SOAP Payload

4.4.1 Separation Information

4.4.1.1 Post Payload

A "Post" is defined (see Section 4.1.1) as sending a request or response to the Broker by a particular connector. This section discusses the (pre-encryption) payloads in the SOAP message.

4.4.1.1.1 State Post Payload

4.4.1.1.1.1 Post to Central Broker Payload

The "Post" payload in the SOAP message is the data defined in the StateSeparationRequestCollection defined in the Separation Request xsd.

```
<?xml version="1.0"?>
<StateSeparationRequestCollection xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
   <StateSeparationRequest>
      <StateRequestRecordGUID>07000000000000000000000000099100</StateRequestR
ecordGUID>
            <SSN>000000546</SSN>
            <ClaimEffectiveDate>2008-11-16</ClaimEffectiveDate>
            <ClaimNumber>0</ClaimNumber>
            <StateEmployerAccountNbr>480616009</StateEmployerAccountNbr>
            <EmployerName>TEAM AUTOMOTIVE LLC</EmployerName>
            <FEIN>841461123</FEIN>
            <TypeofEmployerCode>1</TypeofEmployerCode>
            <TypeofClaimCode>1</TypeofClaimCode>
            <BenefitYearBeginDate>2008-11-16</BenefitYearBeginDate>
            <RequestingStateAbbreviation>CO</RequestingStateAbbreviation>
            <ClaimantLastName>Fortyseven</ClaimantLastName>
            <ClaimantFirstName>Mark</ClaimantFirstName>
            <ClaimantMiddleInitial>A</ClaimantMiddleInitial>
            <ClaimantJobTitle>Test Job Title 47</ClaimantJobTitle>
            <ClaimantReportedFirstDayofWork>2004-05-
17</ClaimantReportedFirstDayofWork>
            <ClaimantReportedLastDayofWork>2008-11-
14</ClaimantReportedLastDayofWork>
            <WagesWeeksNeededCode>NA</WagesWeeksNeededCode>
            <ClaimantSepReasonCode>1</ClaimantSepReasonCode>
            <RequestDate>2008-11-16</RequestDate>
            <ResponseDueDate>2008-12-01</ResponseDueDate>
      </StateSeparationRequest>
      <StateSeparationRequest>
      <StateRequestRecordGUID>07000000000000000000000000099993</StateRequestR
ecordGUID>
```

```
            <SSN>000000510</SSN>
            <ClaimEffectiveDate>2008-11-16</ClaimEffectiveDate>
            <ClaimNumber>0</ClaimNumber>
            <StateEmployerAccountNbr>480616009</StateEmployerAccountNbr>
            <EmployerName>TEAM AUTOMOTIVE LLC</EmployerName>
            <FEIN>841461123</FEIN>
            <TypeofClaimCode>1</TypeofClaimCode>
            <BenefitYearBeginDate>2008-11-16</BenefitYearBeginDate>
            <RequestingStateAbbreviation>CO</RequestingStateAbbreviation>
            <ClaimantLastName>Eleven</ClaimantLastName>
            <ClaimantFirstName>Nate</ClaimantFirstName>
            <ClaimantMiddleInitial>Z</ClaimantMiddleInitial>
            <ClaimantSuffix>III</ClaimantSuffix>
            <ClaimantJobTitle>MaximumCharacTest Job Title
26</ClaimantJobTitle>
            <ClaimantReportedFirstDayofWork>2005-05-
15</ClaimantReportedFirstDayofWork>
            <ClaimantReportedLastDayofWork>2008-11-
14</ClaimantReportedLastDayofWork>
            <WagesWeeksNeededCode>NA</WagesWeeksNeededCode>
            <ClaimantSepReasonCode>99</ClaimantSepReasonCode>
            <RequestDate>2008-11-16</RequestDate>
            <ResponseDueDate>2008-12-01</ResponseDueDate>
      </StateSeparationRequest>
</StateSeparationRequestCollection>
```

4.4.1.1.1.2 Central Broker Acknowledgement to State Payload

In the acknowledgement to the state Post, the Broker sends back a response that contains the number of requests it received, the number in error, and the dates that it started receiving the records and finished receiving the records. This verifies to the state that the Broker received the desired file so it can move on to the next file.

```
<StateSeparationRequestCollectionAcknowledgement xmlns="https:// REDACTED
/schemas">
    <StateRequestFileGUID>D0F7202142A448F0747E99F75CE0FC00</StateRequestFileG
UID>
    <NumberOfRequestRecordsReceived>63</NumberOfRequestRecordsReceived>
    <NumberOfRequestRecordsInError>0</NumberOfRequestRecordsInError>
    <DateStartedReceivingTransmission>2009-07-13T02:37:53.000-
04:00</DateStartedReceivingTransmission>
    <DateFinishedReceivingTransmission>2009-07-13T02:37:54.000-
04:00</DateFinishedReceivingTransmission>
</StateSeparationRequestCollectionAcknowledgement>
```

> ***Note***: The Broker also sends back custom SOAP header information that tells the overall status of the message. This is defined in Section 4.3-SOAP Custom Headers.

If the Broker determines there were Business Rule or XSD errors in the “Post” message, the Broker will also return to the state all of the information that it can on why each individual request failed. The FailedSeparationRequest element defined in the separation request XSD will present the Error Code and the Error Message of the error it found, as described in Part B, Section C-2.8.

```
    <FailedSeparationRequest>
        <StateRequestRecordGUID>00000000000000000000000000099108</StateReques
tRecordGUID>
        <ErrorOccurrence>
            <ErrorCode>101</ErrorCode>
            <ErrorMessage>XSD validation violation</ErrorMessage>
        </ErrorOccurrence>
    </FailedSeparationRequest>
```

Putting this together with the successful acknowledgement:

```
<StateSeparationRequestCollectionAcknowledgement xmlns="https:// REDACTED
/schemas">
    <StateRequestFileGUID>D0F7202142A448F0747E99F75CE0FC00</StateRequestFileG
UID>
    <FailedSeparationRequest>
        <StateRequestRecordGUID>00000000000000000000000000099108</StateReques
tRecordGUID>
        <ErrorOccurrence>
            <ErrorCode>101</ErrorCode>
            <ErrorMessage>XSD validation violation</ErrorMessage>
        </ErrorOccurrence>
    </FailedSeparationRequest>
    <FailedSeparationRequest>
        <StateRequestRecordGUID>00000000000000000000000000099935</StateReques
tRecordGUID>
        <ErrorOccurrence>
            <ErrorCode>101</ErrorCode>
            <ErrorMessage>XSD validation violation</ErrorMessage>
        </ErrorOccurrence>
    </FailedSeparationRequest>
    <FailedSeparationRequest>
        <StateRequestRecordGUID>00000000000000000000000000099999</StateReques
tRecordGUID>
        <ErrorOccurrence>
            <ErrorCode>101</ErrorCode>
            <ErrorMessage>XSD validation violation</ErrorMessage>
        </ErrorOccurrence>
    </FailedSeparationRequest>
    <FailedSeparationRequest>
        <StateRequestRecordGUID>00000000000000000000000000099998</StateReques
tRecordGUID>
        <ErrorOccurrence>
            <ErrorCode>101</ErrorCode>
            <ErrorMessage>XSD validation violation</ErrorMessage>
        </ErrorOccurrence>
    </FailedSeparationRequest>
    <NumberOfRequestRecordsReceived>63</NumberOfRequestRecordsReceived>
    <NumberOfRequestRecordsInError>4</NumberOfRequestRecordsInError>
    <DateStartedReceivingTransmission>2009-07-13T02:37:53.000-
04:00</DateStartedReceivingTransmission>
    <DateFinishedReceivingTransmission>2009-07-13T02:37:54.000-
```

```
04:00</DateFinishedReceivingTransmission>
</StateSeparationRequestCollectionAcknowledgement>
```

4.4.1.1.2 Employer/TPA Post Payload

4.4.1.1.2.1 Post to Central Broker Payload

The “Post” payload in the SOAP message is the data defined in the EmployerTPASeparationResponseCollection defined in the Separation Response xsd

```
<?xml version="1.0"?>
<EmployerTPASeparationResponseCollection xsi:schemaLocation="https://
REDACTED /schemas REDACTED " xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <EmployerTPASeparationResponse>
      <StateRequestRecordGUID>00000000000000000000000000008808</StateRequestR
ecordGUID>
      <BrokerRecordTransactionNumber>2001000</BrokerRecordTransactionNumber>
            <SSN>111111119</SSN>
            <ClaimEffectiveDate>2009-01-04</ClaimEffectiveDate>
            <ClaimNumber>6369857</ClaimNumber>
            <StateEmployerAccountNbr>16475004</StateEmployerAccountNbr>
            <ClaimantNameWorkedAsForEmployer>Gloria Ann
LKJFGRE2</ClaimantNameWorkedAsForEmployer>
            <ClaimantJobTitle>Customer Inquiry Rep</ClaimantJobTitle>
            <SeasonalEmploymentInd>N</SeasonalEmploymentInd>
            <EmployerReportedClaimantFirstDayofWork>2007-11-
05</EmployerReportedClaimantFirstDayofWork>
            <EmployerReportedClaimantLastDayofWork>2008-10-
23</EmployerReportedClaimantLastDayofWork>
            <EffectiveSeparationDate>2008-10-23</EffectiveSeparationDate>
            <TotalEarnedWagesNeededInd>1</TotalEarnedWagesNeededInd>
        <TotalWeeksWorkedNeededInd>1</TotalWeeksWorkedNeededInd>
            <TotalEarnedWages>0</TotalEarnedWages>
            <TotalWeeksWorked>0</TotalWeeksWorked>
      <WagesEarnedAfterClaimEffectiveDate>0</WagesEarnedAfterClaimEffectiveDa
te>
      <NumberOfHoursWorkedAfterClaimEffectiveDate>0</NumberOfHoursWorkedAfter
ClaimEffectiveDate>
            <AverageWeeklyWage>0</AverageWeeklyWage>
            <ClaimantActionsToAvoidQuitInd>N</ClaimantActionsToAvoidQuitInd>
            <EmployerSepReasonCode>6</EmployerSepReasonCode>
            <VoluntarySepReasonCode>1</VoluntarySepReasonCode>
            <ContinuingWorkAvailableInd>Y</ContinuingWorkAvailableInd>
            <VoluntarySepReasonComments>The claimant voluntarily quit without
good cause stating, ‘i loved the time i worked here i had some family
emergencies that were out of my hands so i had some attendance issues but i
will say this is a good company to work for.’  A copy of the claimant’s
online electronically signed Reason for Resignation (Exhibit A) is attached
for your review. The claimant quit without giving a reason. Due to the manner
in which the claimant resigned, we were unable to determine any details
concerning the resignation. We maintain the claimant left for personal
```

```
reasons. We request relief from charges on this
claim.</VoluntarySepReasonComments>
            <PreparerTypeCode>E</PreparerTypeCode>
      <PreparerTelephoneNumberPlusExt>9724312108</PreparerTelephoneNumberPlus
Ext>
            <PreparerContactName>Jay Six</PreparerContactName>
            <PreparerTitle>Project Manager</PreparerTitle>
            <PreparerFaxNbr>9725312108</PreparerFaxNbr>
            <PreparerEmailAddress>j6@jcpenney.com</PreparerEmailAddress>
            <AttachmentID>0</AttachmentID>
      </EmployerTPASeparationResponse>
</EmployerTPASeparationResponseCollection>
```

4.4.1.1.2.2 Central Broker Acknowledgement to Employer/TPA Payload

In the acknowledgement to the state Post, the Broker sends back a response that contains the number of responses it received, the number in error, and the dates that it started receiving the records and finished receiving the records. This verifies to the employer or TPA that the Broker received the desired file so it can move on to the next file.

```
< EmployerTPASeparationResponseCollectionAcknowledgement xmlns="https://
REDACTED /schemas">
    <EmployerTPAResponseFileGUID>542A4AE2395FEDDF1EAA3E57F2DFBCE0</EmployerTP
AResponseFileGUID>
    <NumberOfResponseRecordsReceived>22</NumberOfResponseRecordsReceived>
    <NumberOfResponseRecordsInError>0</NumberOfResponseRecordsInError>
    <DateStartedReceivingTransmission>2009-07-22T03:13:50.000-
04:00</DateStartedReceivingTransmission>
    <DateFinishedReceivingTransmission>2009-07-22T03:13:50.000-
04:00</DateFinishedReceivingTransmission>
</EmployerTPASeparationResponseCollectionAcknowledgement>
```

> ***Note***: The Broker also sends back custom SOAP header information that tells the overall status of the message. This is defined in Section 4.3- SOAP Custom Headers.

If the Broker determines there were Business Rule or XSD errors in the “Post” message, the Broker will also return to the employer or TPA all of the information that it can on why each individual response failed. The FailedSeparationResponse element defined in the separation response XSD will present the Error Code and the Error Message of the error it found, as described in Part B, Section C-2.8.

```
    <FailedSeparationResponse>

<StateRequestRecordGUID>00000000000000000000000000099943</StateRequestRecordG
UID>
        <BrokerRecordTransactionNumber>2001552</BrokerRecordTransactionNumber
>
        <ErrorOccurrence>
            <ErrorCode>213</ErrorCode>
            <ErrorMessage>Business Rule violation - There must be a value
(Character - Size 1) for TotalWeeksWorkedNeededInd if WagesWeeksNeededCode =
WW</ErrorMessage>
        </ErrorOccurrence>
    </FailedSeparationResponse>
```

Putting this together with the successful acknowledgement:

```
<EmployerTPASeparationResponseCollectionAcknowledgement xmlns="https://
REDACTED /schemas">
    <EmployerTPAResponseFileGUID>542A4AE2395FEDDF1EAA3E57F2DFBCE0</EmployerTP
AResponseFileGUID>

  <FailedSeparationResponse>
        <StateRequestRecordGUID>00000000000000000000000000099943</StateReques
tRecordGUID>
        <BrokerRecordTransactionNumber>2001552</BrokerRecordTransactionNumber
>
        <ErrorOccurrence>
            <ErrorCode>213</ErrorCode>
            <ErrorMessage>Business Rule violation - There must be a value
(Character - Size 1) for TotalWeeksWorkedNeededInd if WagesWeeksNeededCode =
WW</ErrorMessage>
        </ErrorOccurrence>
    </FailedSeparationResponse>
    <NumberOfResponseRecordsReceived>22</NumberOfResponseRecordsReceived>
    <NumberOfResponseRecordsInError>1</NumberOfResponseRecordsInError>
    <DateStartedReceivingTransmission>2009-07-22T03:13:50.000-
04:00</DateStartedReceivingTransmission>
    <DateFinishedReceivingTransmission>2009-07-22T03:13:50.000-
04:00</DateFinishedReceivingTransmission>
</EmployerTPASeparationResponseCollectionAcknowledgement>
```

4.4.1.2 Pull Payload

A “Pull” is defined (see Section 4.1.2) as sending a query to the **Central Broker** to allow the **Central Broker** to send the connector all of the requests or responses it has waiting for it (in multiple transactions if they are from different endpoints). This section discusses the (pre-encryption) payloads in the SOAP message.

4.4.1.2.1 State Pull Payload

4.4.1.2.1.1 Pull from Central Broker

The StateSeparationResponseCollectionQuery defined in **REDACTED** WSDL (see Section 4.6) is a complex type that allows the caller to specify one of three operations: a Pull, a Re-Pull by StateSOAPTransactionNumber, and a Re-Pull by a Date Range.

```
<!-- Query element for states to collect claim responses they are expecting -
->
    <xs:element name=" REDACTED "
        type REDACTED "/>

    <!-- Types for query element for states to collect claim responses they
are expecting -->
```

```
    <xs:complexType name="StateSeparationResponseCollectionQueryType">
        <xs:sequence>
            <xs:element name="StatePostalCode" type="StateAbrCodes" />
            <xs:element name="StateSeparationResponseCollectionQueryCriteria"
                type="StateSeparationResponseCollectionQueryCriteriaType"
                minOccurs="0" />
        </xs:sequence>
        <xs:attribute ref="wsu:Id" use="optional"/>
    </xs:complexType>

    <xs:complexType
name="StateSeparationResponseCollectionQueryCriteriaType">
        <xs:sequence>
              <xs:element name="StateSOAPTransactionNumber"
type="xs:nonNegativeInteger" minOccurs="0"/>
              <xs:group
ref="StateSeparationResponseCollectionQueryCriteriaGroup" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:group name="StateSeparationResponseCollectionQueryCriteriaGroup">
        <xs:sequence>
            <xs:element name="BrokerRecordEffectiveDateFrom"
type="CustomDateTime" />
            <xs:element name="BrokerRecordEffectiveDateTo"
type="CustomDateTime" />
        </xs:sequence>

    </xs:group>
```

For the straight Pull, the caller needs to supply only the state Unique ID. Although there are different ways to verify the calling state besides this element, the Broker uses it as an additional security check. Also, there is a requirement in the WSDL 1.1 specification that a WSDL definition have at least one input attribute.

```
<?xml version="1.0" encoding="US-ASCII"?>
<StateSeparationResponseCollectionQuery xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas SeparationResponse.xsd ">
  <StatePostalCode>ST</StatePostalCode>
</StateSeparationResponseCollectionQuery>
```

For the Re-Pull by StateSOAPTransactionNumber, the caller needs to supply the state Unique ID and the StateSOAPTransactionNumber element out of the StateSeparationResponseCollectionQueryCriteriaType. This will allow the Broker to send the file defined by the StateSOAPTransactionNumber.

```
<?xml version="1.0" encoding="US-ASCII"?>
<StateSeparationResponseCollectionQuery xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">
  <StatePostalCode>ST</StatePostalCode>
  <StateSeparationResponseCollectionQueryCriteria>
    <StateSOAPTransactionNumber>123456221</StateSOAPTransactionNumber>
  </StateSeparationResponseCollectionQueryCriteria>
</StateSeparationResponseCollectionQuery>
```

For the Re-Pull by date range, the caller needs to supply the state Unique ID and the StateSeparationResponseCollectionQueryCriteriaGroup element out of the StateSeparationResponseCollectionQueryCriteriaType The Re-Pull by date range will pull all the files that were pulled by the connector during the date range specified.

The StateSeparationResponseCollectionQueryCriteriaGroup is a complex type that is defined as a begin date (BrokerRecordEffectiveDateFrom), an end date (BrokerRecordEffectiveDateTo) and a StateSOAPTransactionNumber.

The first time this operation is called, the StateSOAPTransactionNumber must not be included and the date range that the files to be Re-Pulled are included.

WARNING: If the end date in the Re-Pull by date range is in the future, this will cause the **Central Broker** to resend all transactions ***including*** all the resent transactions that the **Central Broker** has been delivering due to this call, thus putting your Connector into an infinite loop until that date is reached. This will tax the Connector and the **Central Broker** needlessly and must be avoided.

When the Broker sends back the first file in this date range, it will include in the SOAP header the next StateSOAPTransactionNumber that it sent during that date range (in element name NextStateSOAPTransactionNumber). In the next call to this operation, the caller must include the NextStateSOAPTransactionNumber as the StateSOAPTransactionNumber along with the date range. This differentiates to the Broker the next call in the series from a brand new Re-Pull by Date request.

When the **Central Broker** determines that it has no more files to send back to the connector in the given date range, the last file sent back to the connector is indicated by the **Central Broker** not including the next StateSOAPTransactionNumber (so there will not be a NextStateSOAPTransactionNumber included in the http response SOAP header).

First Call:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!-- test query to re-pull by date range, pulls up to 8mb of records -->
<StateSeparationResponseCollectionQuery xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">
  <StatePostalCode>ST</StatePostalCode>
  <StateSeparationResponseCollectionQueryCriteria>
    <BrokerRecordEffectiveDateFrom>2009-01-01T12:00:00.000-
04:00</BrokerRecordEffectiveDateFrom>
    <BrokerRecordEffectiveDateTo>2009-12-31T12:00:00.000-
04:00</BrokerRecordEffectiveDateTo>
  </StateSeparationResponseCollectionQueryCriteria>
</StateSeparationResponseCollectionQuery>
```

Subsequent Calls:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!-- test query to re-pull by date range, pulls up to 8mb of records -->
<StateSeparationResponseCollectionQuery xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">
  <StatePostalCode>ST</StatePostalCode>
  <StateSeparationResponseCollectionQueryCriteria>
      <StateSOAPTransactionNumber>12345678901234567890123456789012</StateSOAP
TransactionNumber>
      <BrokerRecordEffectiveDateFrom>2009-01-01T12:00:00.000-
04:00</BrokerRecordEffectiveDateFrom>
      <BrokerRecordEffectiveDateTo>2009-12-31T12:00:00.000-
04:00</BrokerRecordEffectiveDateTo>
  </StateSeparationResponseCollectionQueryCriteria>
</StateSeparationResponseCollectionQuery>
```

4.4.1.2.1.2 Central Broker Response to the State

When the Broker receives a “Pull” request from a State, it begins assembling all the responses that are intended for that state. It constructs a SOAP message according to the rules for a Separation Information SOAP message (less than 8MB, one employer or TPA per message, etc.). It adds the additional field BrokerRecordEffectiveDate to the response, which indicates the date that it was accepted into the Broker. It then sends the responses in the HTTP response.

```
<StateSeparationResponseCollection xmlns="https:// REDACTED /schemas">
    <StateSeparationResponse>
        <StateRequestRecordGUID>000000000000000000000000000099100</StateReques
tRecordGUID>
        <BrokerRecordTransactionNumber>2001636</BrokerRecordTransactionNumber
>
        <SSN>000000546</SSN>
        <ClaimEffectiveDate>2008-11-16</ClaimEffectiveDate>
        <ClaimNumber>0</ClaimNumber>
        <StateEmployerAccountNbr>480616009</StateEmployerAccountNbr>
        <ClaimantJobTitle>Test Job Title 47</ClaimantJobTitle>
        <SeasonalEmploymentInd>N</SeasonalEmploymentInd>
        <EmployerReportedClaimantFirstDayofWork>2004-05-
17</EmployerReportedClaimantFirstDayofWork>
        <EmployerReportedClaimantLastDayofWork>2008-11-
14</EmployerReportedClaimantLastDayofWork>
        <AverageWeeklyWage>1575.33</AverageWeeklyWage>
        <EmployerSepReasonCode>1</EmployerSepReasonCode>
        <ReturnToWorkInd>Y</ReturnToWorkInd>
        <ReturnToWorkDate>2009-02-01</ReturnToWorkDate>
        <Remuneration>
            <RemunerationTypeCode>1</RemunerationTypeCode>
            <RemunerationAmountPerPeriod>393.83</RemunerationAmountPerPeriod>
            <RemunerationPeriodFrequencyCode>W</RemunerationPeriodFrequencyCo
```

```
de>
            <DateRemunerationIssued>2008-11-14</DateRemunerationIssued>
            <EmployerAllocationInd>Y</EmployerAllocationInd>
            <AllocationBeginDate>2008-11-14</AllocationBeginDate>
            <AllocationEndDate>2008-12-14</AllocationEndDate>
        </Remuneration>
        <AverageNumberofHoursWorkedperWeek>40</AverageNumberofHoursWorkedperW
eek>
        <EmployerSepReasonComments>Test Sep Reason Comments
Code1</EmployerSepReasonComments>
        <AttachmentOccurrence>
            <UniqueAttachmentId>1</UniqueAttachmentId>
            <DescriptionofAttachmentCode>1</DescriptionofAttachmentCode>
            <TypeofDocument>test type of document</TypeofDocument>
            <AttachmentSize>2</AttachmentSize>
            <AttachmentData>UjBsR09EbGhjZ0dTQUxNQUFBUUNBRU1tQ1p0dU1GUXhEUzhi<
/AttachmentData>
        </AttachmentOccurrence>
        <PreparerTypeCode>T</PreparerTypeCode>
        <PreparerCompanyName>ADP</PreparerCompanyName>
        <PreparerTelephoneNumberPlusExt>4445557777</PreparerTelephoneNumberPl
usExt>
        <PreparerContactName>Preparer ADP Name Maximum Character Testing
Sixty Characte</PreparerContactName>
        <PreparerTitle>Preparer ADP Title Max Charact</PreparerTitle>
        <PreparerFaxNbr>4445557788</PreparerFaxNbr>
        <PreparerEmailAddress>adppreparer@test.com</PreparerEmailAddress>
        <BrokerRecordEffectiveDate>2009-07-22T01:39:20.000-
04:00</BrokerRecordEffectiveDate>
    </StateSeparationResponse>
</StateSeparationResponseCollection>
```

4.4.1.2.1.3 State Acknowledgement to the Central Broker

The StateSeparationResponseCollectionAcknowledgement is initiated once the state has received its file from the Broker. The acknowledgement must accompany every state Pull request, as this is the manner in which the Broker knows that the state Pull was successful. This is required even if the Broker has sent back an empty file and a MessageCode of 2. If this is not sent back to the Broker, the next “Pull” call to the Broker will result in the same file being passed back. The Broker will not move on to the next file until it receives a successful acknowledgement. If the Broker receives 3 unanswered Pull requests, it will suspend any processing of Pull requests by the State until the **Broker Administrator** and State Administrator can work out the problem.

The key field in this message is the StateSOAPTransmissionNumber, which must correspond with the StateSOAPTransmissionNumber sent back in the Broker Response. The remainder of the message is just reporting information; the values are not used for anything at this time. If the state does not collect this information, just return 0 for the number of records and place a valid date in the date fields.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<StateSeparationResponseCollectionAcknowledgement xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">

<StateSOAPTransmissionNumber>12345678901234567890123456789012</StateSOAPTrans
missionNumber>
  <NumberOfResponseRecordsReceived>0</NumberOfResponseRecordsReceived>
  <NumberOfResponseRecordsInError>0</NumberOfResponseRecordsInError>
  <DateStartedReceivingTransmission>2008-12-31T12:00:00.000-
04:00</DateStartedReceivingTransmission>
  <DateFinishedReceivingTransmission>2008-12-31T12:00:00.000-
04:00</DateFinishedReceivingTransmission>
</StateSeparationResponseCollectionAcknowledgement>
```

4.4.1.2.2 Employer/TPA Pull

4.4.1.2.2.1 Pull from Central Broker

The EmployerTPASeparationRequestCollectionQuery is a complex query type that allows the caller to specify one of three operations: a Pull, a Re-Pull by EmployerTPASOAPTransactionNumber, and a Re-Pull by a Date Range.

```
 <!-- Query element for employer to collect claim responses they are
expecting -->
    <xs:element name=" REDACTED "
        type=" REDACTED "/>

    <!-- Types for query element for Employers/TPAs to collect claim requests
they are expecting -->
    <xs:complexType name="EmployerTPASeparationRequestCollectionQueryType">
        <xs:sequence>
            <xs:element name="UniqueID" type="UniqueIDType" />
            <xs:element
name="EmployerTPASeparationRequestCollectionQueryCriteria"

type="EmployerTPASeparationRequestCollectionQueryCriteriaType"
                minOccurs="0" />
        </xs:sequence>
        <xs:attribute ref="wsu:Id" use="optional"/>
    </xs:complexType>

    <xs:complexType
name="EmployerTPASeparationRequestCollectionQueryCriteriaType">
        <xs:sequence>
              <xs:element name="EmployerTPASOAPTransactionNumber"
type="xs:nonNegativeInteger" minOccurs="0" />
            <xs:group
ref="EmployerTPASeparationRequestCollectionQueryCriteriaGroup"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:group
name="EmployerTPASeparationRequestCollectionQueryCriteriaGroup">
        <xs:sequence>
            <xs:element name="BrokerRecordEffectiveDateFrom"
type="CustomDateTime" />
            <xs:element name="BrokerRecordEffectiveDateTo"
type="CustomDateTime" />
        </xs:sequence>

    </xs:group>
```

For the straight Pull, the caller needs to supply only the employer or TPA Unique ID. Although there are different ways to verify the calling employer or TPA besides this element, the Broker uses it as an additional security check. Also, there is a requirement in WSDL 1.1 that a WSDL definition have at least one input attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPASeparationRequestCollectionQuery xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">
  <UniqueID>BR000000001</UniqueID>
</EmployerTPASeparationRequestCollectionQuery>
```

For the Re-Pull by EmployerTPASOAPTransactionNumber, the caller needs to supply the Employer/TPA Unique ID and the EmployerTPASOAPTransactionNumber element out of the EmployerTPASeparationResponseCollectionQueryCriteriaType. This will allow the Broker to send the file defined by the EmployerTPASOAPTransactionNumber.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPASeparationRequestCollectionQuery xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">
  <UniqueID>BR000000001</UniqueID>
  <EmployerTPASeparationRequestCollectionQueryCriteria>
      <EmployerTPASOAPTransactionNumber>26151</EmployerTPASOAPTransactionNumb
er>
  </EmployerTPASeparationRequestCollectionQueryCriteria>
</EmployerTPASeparationRequestCollectionQuery>
```

For the Re-Pull by date range, the caller needs to supply the EmployerTPA Unique ID and the EmployerTPASeparationResponseCollectionQueryCriteriaGroup element out of the EmployerTPASeparationResponseCollectionQueryCriteriaType. The Re-Pull by date range will pull all the files that were pulled by the connector during the date range specified.

The EmployerTPASeparationResponseCollectionQueryCriteriaGroup is a complex query type that is defined as a begin date (BrokerRecordEffectiveDateFrom), an end date

WARNING: If the end date in the Re-Pull by date range is in the future, this will cause the **Central Broker** to resend all transactions ***including*** all the resent transactions that the **Central Broker** has been delivering due to this call, thus putting your Connector into an infinite loop until that date is reached. This will tax the Connector and the **Central Broker** needlessly and must be avoided.

(BrokerRecordEffectiveDateTo) and an EmployerTPASOAPTransactionNumber.

The first time this operation is called, the EmployerTPASOAPTransactionNumber must not be included and the date range that the files to be Re-Pulled are included.

When the Broker sends back the first file in this date range, it will include in the SOAP header the next EmployerTPASOAPTransactionNumber that it sent during that date range (in element name NextEmployerTPASOAPTransactionNumber). In the next call to this operation, the caller must include the NextEmployerTPASOAPTransactionNumber as the EmployerTPASOAPTransactionNumber along with the date range. This differentiates to the Broker the next call in the series from a brand new Re-Pull by Date request.

When the **Central Broker** determines that it has no more files to send back to the connector in the given date range, the last file sent back to the connector is indicated by the **Central Broker** not including the next EmployerTPASOAPTransactionNumber (so there will not be a NextEmployerTPASOAPTransactionNumber included in the http response SOAP header).

First Call:

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPASeparationRequestCollectionQuery xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">
  <UniqueID>BR000000001</UniqueID>
  <EmployerTPASeparationRequestCollectionQueryCriteria>
    <BrokerRecordEffectiveDateFrom>2009-01-01T12:00:00.000-
04:00</BrokerRecordEffectiveDateFrom>
    <BrokerRecordEffectiveDateTo>2009-12-31T12:00:00.000-
04:00</BrokerRecordEffectiveDateTo>
  </EmployerTPASeparationRequestCollectionQueryCriteria>
</EmployerTPASeparationRequestCollectionQuery>
```

Subsequent Calls:

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPASeparationRequestCollectionQuery xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">
  <UniqueID>BR000000001</UniqueID>
  <EmployerTPASeparationRequestCollectionQueryCriteria>
      <EmployerTPASOAPTransactionNumber>12345678901234567890123456789012</Emp
loyerTPASOAPTransactionNumber>
      <BrokerRecordEffectiveDateFrom>2009-01-01T12:00:00.000-
04:00</BrokerRecordEffectiveDateFrom>
      <BrokerRecordEffectiveDateTo>2009-12-31T12:00:00.000-
04:00</BrokerRecordEffectiveDateTo>
  </EmployerTPASeparationRequestCollectionQueryCriteria>
</EmployerTPASeparationRequestCollectionQuery>
```

4.4.1.2.2.2 Central Broker Response to Employer/TPA

When the Broker receives a “Pull” request from an Employer/TPA, it begins assembling all the requests that are intended for that employer or TPA. It constructs a SOAP message according to the rules for a Separation Information SOAP message (less than 8MB, one employer or TPA per message, etc.). It adds two additional fields to the Separation Request - the BrokerRecordEffectiveDate and the BrokerRecordTransactionNumber. The BrokerRecordEffectiveDate indicates the date that it was accepted into the Broker. The BrokerRecordTransactionNumber is a unique record tracking number and must be returned on the response for this record. It then sends the separation requests in the HTTP response.

```
<EmployerTPASeparationRequestCollection xmlns="https:// REDACTED /schemas">
    <EmployerTPASeparationRequest>
        <StateRequestRecordGUID>00000000000000000000000000099960</StateReques
tRecordGUID>
        <SSN>000000618</SSN>
        <ClaimEffectiveDate>2008-11-23</ClaimEffectiveDate>
        <ClaimNumber>1</ClaimNumber>
        <StateEmployerAccountNbr>555444333</StateEmployerAccountNbr>
        <EmployerName>JC Penney</EmployerName>
        <FEIN>123456789</FEIN>
        <TypeofEmployerCode>4</TypeofEmployerCode>
        <TypeofClaimCode>1</TypeofClaimCode>
        <BenefitYearBeginDate>2008-11-23</BenefitYearBeginDate>
        <RequestingStateAbbreviation>CO</RequestingStateAbbreviation>
        <ClaimantLastName>Sixhundredeighteen</ClaimantLastName>
        <ClaimantFirstName>William</ClaimantFirstName>
        <ClaimantMiddleInitial>R</ClaimantMiddleInitial>
        <ClaimantJobTitle>Test Job Title 618</ClaimantJobTitle>
        <ClaimantReportedFirstDayofWork>2006-11-
17</ClaimantReportedFirstDayofWork>
        <ClaimantReportedLastDayofWork>2008-11-
21</ClaimantReportedLastDayofWork>
        <WagesWeeksNeededCode>NA</WagesWeeksNeededCode>
        <ClaimantSepReasonCode>18</ClaimantSepReasonCode>
        <RequestDate>2008-11-23</RequestDate>
        <ResponseDueDate>2008-12-08</ResponseDueDate>
        <BrokerRecordTransactionNumber>2001569</BrokerRecordTransactionNumber
>
        <BrokerRecordEffectiveDate>2009-07-13T14:35:58.000-
04:00</BrokerRecordEffectiveDate>
    </EmployerTPASeparationRequest>
</EmployerTPASeparationRequestCollection>
```

4.4.1.2.2.3 Employer/TPA Acknowledgement to Central Broker

The EmployerTPASeparationRequestCollectionAcknowledgement is initiated once the employer or TPA has received its file from the Broker. The acknowledgement must accompany every employer or TPA Pull request, as this is the manner in which the Broker knows that the Employer/TPA Pull was successful. This is required even if the Broker has sent back an empty file and a MessageCode of 2. If this is not sent back to the Broker, the next “Pull” call to the Broker will result in the same file being passed back. The Broker will not move on to the next file until it receives a successful acknowledgement. If the Broker receives 3 unanswered Pull requests, it will suspend any processing of Pull requests by the Employer/TPA until the **Broker Administrator** and Employer/TPA Administrator can work out the problem.

The key field in this message is the EmployerTPASOAPTransmissionNumber, which must correspond with the EmployerTPASOAPTransmissionNumber sent back in the Broker Response. The remainder of the message is just reporting information; the values are not used for anything at this time. If the EmployerTPA does not collect this information, just return 0 for the number of records and place a valid date in the date fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPASeparationRequestCollectionAcknowledgement xmlns="https://
REDACTED /schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas combined.xsd ">
  <EmployerTPASOAPTransmissionNumber>3211</EmployerTPASOAPTransmissionNumber>
  <NumberOfRequestRecordsReceived>4</NumberOfRequestRecordsReceived>
  <NumberOfRequestRecordsInError>0</NumberOfRequestRecordsInError>
  <DateStartedReceivingTransmission>2001-12-31T12:00:00.000-
04:00</DateStartedReceivingTransmission>
  <DateFinishedReceivingTransmission>2001-12-31T12:00:00.000-
04:00</DateFinishedReceivingTransmission>
</EmployerTPASeparationRequestCollectionAcknowledgement>
```

4.4.2 Earnings Verification

4.4.2.1 Post Payload

A “Post” is defined (see Section 4.1.1) as sending a request or response to the Broker by a particular connector. This section discusses the (pre-encryption) payloads in the SOAP message for the Earnings Verification Exchange.

4.4.2.1.1 State Post Payload

4.4.2.1.1.1 Post to Central Broker Payload

The “Post” payload in the SOAP message is the data defined in the StateEarningsVerificationRequestCollection defined in the Earnings Verification Request xsd.

```
<?xml version="1.0"?>
<StateEarningsVerificationRequestCollection xsi:schemaLocation="https://
REDACTED /schemas EarningsVerificationRequest.xsd" xmlns="https://
REDACTED /schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <StateEarningsVerificationRequest>
      <StateEarningsVerificationRequestRecordGUID>AAA5300000000000000000000
00003</StateEarningsVerificationRequestRecordGUID>
      <RequestingStateAbbreviation>ST</RequestingStateAbbreviation>
      <UIOfficeName>Office Name</UIOfficeName>
      <UIOfficePhone>5555555555</UIOfficePhone>
      <UIOfficeFax>5555555554</UIOfficeFax>
      <UIOfficeEmailAddress>james.madison@state.gov</UIOfficeEmailAddress>
      <StateEmployerAccountNbr>1234567890</StateEmployerAccountNbr>
      <FEIN>123456789</FEIN>
      <EmployerName>ACME</EmployerName>
      <SSN>311111334</SSN>
      <ClaimantLastName>Lastname</ClaimantLastName>
      <ClaimantFirstName>Firstname</ClaimantFirstName>
      <ClaimantMiddleInitial>M</ClaimantMiddleInitial>
      <ClaimantSuffix>JR</ClaimantSuffix>
      <NumberofWeeksRequested>5</NumberofWeeksRequested>
      <EarningsVerificationWeekBeginDate>2010-08-
01</EarningsVerificationWeekBeginDate>
      <EarningsVerificationWeekEndDate>2010-09-
04</EarningsVerificationWeekEndDate>
      <EarningsVerificationComments>This is a comment field for this Earnings
Verification Request</EarningsVerificationComments>
      <RequestDate>2010-10-14</RequestDate>
      <EarningsStatusCode>3</EarningsStatusCode>
      <TipsStatusCode>1</TipsStatusCode>
      <CommissionStatusCode>1</CommissionStatusCode>
      <BonusStatusCode>1</BonusStatusCode>
      <VacationStatusCode>1</VacationStatusCode>
      <SickLeaveStatusCode>1</SickLeaveStatusCode>
      <HolidayStatusCode>3</HolidayStatusCode>
      <SeveranceStatusCode>3</SeveranceStatusCode>
      <WagesInLieuStatusCode>4</WagesInLieuStatusCode>
      <EarningsVerificationResponseCommentIndicator>1</EarningsVerificationRe
sponseCommentIndicator>
      <ResponseDueDate>2010-10-28</ResponseDueDate>
      <EarningsVerificationSourceCode>9</EarningsVerificationSourceCode>
   </StateEarningsVerificationRequest>

</StateEarningsVerificationRequestCollection>
```

4.4.2.1.1.2 Central Broker Acknowledgement to State Payload

In the acknowledgement to the state Post, the Broker sends back a response that contains the number of requests it received, the number in error, and the dates that it started receiving the records and finished receiving the records. This verifies to the state that the Broker received the desired file so it can move on to the next file.

```
<StateEarningsVerificationRequestCollectionAcknowledgement xmlns="https://
REDACTED /schemas">
    <StateRequestFileGUID>C0EB9C5D24CA4B8EFB8AEED97A5252C8</StateRequestFileG
UID>
    <NumberOfRequestRecordsReceived>1</NumberOfRequestRecordsReceived>
    <NumberOfRequestRecordsInError>0</NumberOfRequestRecordsInError>
    <DateStartedReceivingTransmission>2010-11-18T10:26:52.730-
05:00</DateStartedReceivingTransmission>
    <DateFinishedReceivingTransmission>2010-11-18T10:26:54.610-
05:00</DateFinishedReceivingTransmission>
</StateEarningsVerificationRequestCollectionAcknowledgement>
```

> ***Note***: The Broker also sends back custom SOAP header information that tells the overall status of the message. This is defined in Section 4.3- SOAP Custom Headers.

If the Broker determines there were Business Rule or XSD errors in the "Post" message, the Broker will also return to the state all of the information that it can on why each individual request failed. The FailedEarningVerificationRequest element defined in the earnings verification request XSD will present the Error Code and the Error Message of the error it found, as described in Part B, Section C-2.8.

```
     <FailedEarningsVerificationRequest>
        <StateEarningsVerificationRequestRecordGUID>31101000000000000000000
0000001</StateEarningsVerificationRequestRecordGUID>
        <ErrorOccurrence>
            <ErrorCode>311</ErrorCode>
            <ErrorMessage>Business Rule violation - The
NumberofWeeksRequested (Character - Size 2) must equal the number of days
included in EarningsVerificationWeekBeginDate thru
EarningsVerificationWeekEndDate divided by 7 days.</ErrorMessage>
        </ErrorOccurrence>
    </FailedEarningsVerificationRequest>
```

Putting this together with the successful acknowledgement:

```
<StateEarningsVerificationRequestCollectionAcknowledgement xmlns="https://
REDACTED /schemas">
    <StateRequestFileGUID>9CEBA17833DFD7C6476EEDE25D20FFF6</StateRequestFileG
UID>
    <FailedEarningsVerificationRequest>
        <StateEarningsVerificationRequestRecordGUID>31101000000000000000000
0000001</StateEarningsVerificationRequestRecordGUID>
        <ErrorOccurrence>
            <ErrorCode>311</ErrorCode>
            <ErrorMessage>Business Rule violation - The
NumberofWeeksRequested (Character - Size 2) must equal the number of days
included in EarningsVerificationWeekBeginDate thru
EarningsVerificationWeekEndDate divided by 7 days.</ErrorMessage>
        </ErrorOccurrence>
    </FailedEarningsVerificationRequest>
    <NumberOfRequestRecordsReceived>1</NumberOfRequestRecordsReceived>
    <NumberOfRequestRecordsInError>1</NumberOfRequestRecordsInError>
    <DateStartedReceivingTransmission>2011-01-21T10:49:27.929-
05:00</DateStartedReceivingTransmission>
    <DateFinishedReceivingTransmission>2011-01-21T10:49:28.181-
05:00</DateFinishedReceivingTransmission>
</StateEarningsVerificationRequestCollectionAcknowledgement>
```

4.4.2.1.2 Employer/TPA Post Payload

4.4.2.1.2.1 Post to Central Broker Payload

The “Post” payload in the SOAP message is the data defined in the EmployerTPAEarningsVerificationResponseCollection defined in the Earnings Verification Response xsd.

```
<?xml version="1.0"?>
<EmployerTPAEarningsVerificationResponseCollection
xsi:schemaLocation="https:// REDACTED /schemas
EarningsVerificationResponse.xsd" xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <EmployerTPAEarningsVerificationResponse>
      <!-- Backfilled -->
      <StateEarningsVerificationRequestRecordGUID>AAA5300000000000000000000
00003</StateEarningsVerificationRequestRecordGUID>
      <!-- Backfilled -->
      <BrokerRecordTransactionNumber>5447</BrokerRecordTransactionNumber>
      <!-- Backfilled -->
      <RequestingStateAbbreviation>ST</RequestingStateAbbreviation>
      <!-- Backfilled -->
      <UIOfficeName>Office Name</UIOfficeName>
      <!-- Backfilled -->
      <StateEmployerAccountNbr>1234567890</StateEmployerAccountNbr>
      <!-- Backfilled -->
      <FEIN>123456789</FEIN>
      <CorrectedFEIN>987654321</CorrectedFEIN>
      <!-- Backfilled -->
      <EmployerName>ACME</EmployerName>
      <CorrectedEmployerName>Fly By Night</CorrectedEmployerName>
      <!-- Backfilled -->
      <SSN>311111334</SSN>
      <ClaimantNameWorkedAsForEmployer>John Q
Public</ClaimantNameWorkedAsForEmployer>
      <!-- Backfilled -->
      <NumberofWeeksRequested>5</NumberofWeeksRequested>
      <!-- Backfilled -->
      <EarningsVerificationWeekBeginDate>2010-08-
01</EarningsVerificationWeekBeginDate>
      <!-- Backfilled -->
      <EarningsVerificationWeekEndDate>2010-09-
04</EarningsVerificationWeekEndDate>
      <!-- 1 - Claimaint works, 20 - Never Employed Here, 21 - TPA does not
represent Employer -->
      <ClaimantEmployerWorkRelationshipCode>1</ClaimantEmployerWorkRelationsh
ipCode>
      <!-- 1 Yes, has earnings, 2 - did not have earnings (100% Sales), 9 -
No Work -->
      <EmployerEarningsCode>1</EmployerEarningsCode>
      <FirstDayWorkedinPeriod>2010-08-01</FirstDayWorkedinPeriod>
      <!-- 1 for Yes, 2 for No -->
      <StillWorkingCode>2</StillWorkingCode>
      <LastDayWorked>2010-09-04</LastDayWorked>
      <!-- 1 - Layoff, 2 - Fired, 3 - Vol Quit, 4 - Other -->
      <EmployerSepReasonCode>1</EmployerSepReasonCode>
      <!-- When Request = 1 or (2 with Work/Relationship = 20/21 or Earnings
Code = 9) -->
```

```
      <EarningsVerificationResponseComment>This employee was let go during
the time period</EarningsVerificationResponseComment>
      <WeeklyEarningsVerification>
        <WeekBeginDate>2010-08-01</WeekBeginDate>
        <WeekEndDate>2010-08-07</WeekEndDate>
        <HoursWorked>15:00</HoursWorked>
        <!-- See Request values for required or not for all below -->
        <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        <EarningsPaidDate>2010-08-21</EarningsPaidDate>
        <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
        <HolidayPaidDate>2010-08-07</HolidayPaidDate>
        <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
        <SeverancePaidDate>2010-08-07</SeverancePaidDate>
        <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
        <WagesInLieuPaidDate>2010-08-07</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <WeeklyEarningsVerification>
        <WeekBeginDate>2010-08-08</WeekBeginDate>
        <WeekEndDate>2010-08-14</WeekEndDate>
        <HoursWorked>15:00</HoursWorked>
        <!-- See Request values for required or not for all below -->
        <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        <EarningsPaidDate>2010-08-21</EarningsPaidDate>
        <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
        <HolidayPaidDate>2010-08-14</HolidayPaidDate>
        <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
        <SeverancePaidDate>2010-08-14</SeverancePaidDate>
        <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
        <WagesInLieuPaidDate>2010-08-14</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <WeeklyEarningsVerification>
        <WeekBeginDate>2010-08-15</WeekBeginDate>
        <WeekEndDate>2010-08-21</WeekEndDate>
        <HoursWorked>15:00</HoursWorked>
        <!-- See Request values for required or not for all below -->
        <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        <EarningsPaidDate>2010-08-21</EarningsPaidDate>
        <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
        <HolidayPaidDate>2010-08-21</HolidayPaidDate>
        <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
        <SeverancePaidDate>2010-08-21</SeverancePaidDate>
        <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
        <WagesInLieuPaidDate>2010-08-21</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <WeeklyEarningsVerification>
        <WeekBeginDate>2010-08-22</WeekBeginDate>
        <WeekEndDate>2010-08-28</WeekEndDate>
        <HoursWorked>101:00</HoursWorked>
        <!-- See Request values for required or not for all below -->
        <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        <EarningsPaidDate>2010-08-28</EarningsPaidDate>
        <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
        <HolidayPaidDate>2010-08-28</HolidayPaidDate>
        <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
        <SeverancePaidDate>2010-08-28</SeverancePaidDate>
        <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
        <WagesInLieuPaidDate>2010-08-28</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <WeeklyEarningsVerification>
        <WeekBeginDate>2010-08-29</WeekBeginDate>
        <WeekEndDate>2010-09-04</WeekEndDate>
        <HoursWorked>5:00</HoursWorked>
        <!-- See Request values for required or not for all below -->
        <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        <EarningsPaidDate>2010-09-04</EarningsPaidDate>
        <HolidayAmountPaidForWeek>60.00</HolidayAmountPaidForWeek>
        <HolidayPaidDate>2010-09-04</HolidayPaidDate>
        <SeveranceAmountPaidForWeek>70.00</SeveranceAmountPaidForWeek>
        <SeverancePaidDate>2010-09-04</SeverancePaidDate>
        <WagesInLieuAmountPaidForWeek>80.00</WagesInLieuAmountPaidForWeek>
        <WagesInLieuPaidDate>2010-09-04</WagesInLieuPaidDate>
      </WeeklyEarningsVerification>
      <!-- E - Employer, T - TPA -->
      <PreparerTypeCode>T</PreparerTypeCode>
      <PreparerCompanyName>ABC TPA</PreparerCompanyName>
      <PreparerTelephoneNumberPlusExt>5555555556</PreparerTelephoneNumberPlus
Ext>
      <PreparerContactName>Mrs Sue Herman</PreparerContactName>
      <PreparerTitle>Claims Administrator</PreparerTitle>
      <PreparerFaxNbr>5555555557</PreparerFaxNbr>
      <PreparerEmailAddress>sue.herman@abctpa.com</PreparerEmailAddress>
      <!-- Backfilled -->
      <EarningsVerificationSourceCode>9</EarningsVerificationSourceCode>
  </EmployerTPAEarningsVerificationResponse>
</EmployerTPAEarningsVerificationResponseCollection>
```

4.4.2.1.2.2 Central Broker Acknowledgement to Employer/TPA Payload

In the acknowledgement to the state Post, the Broker sends back a response that contains the number of responses it received, the number in error, and the dates that it started receiving the records and finished receiving the records. This verifies to the employer or TPA that the Broker received the desired file so it can move on to the next file.

```
<EmployerTPAEarningsVerificationResponseCollectionAcknowledgement
xmlns="https:// REDACTED /schemas">
    <EmployerTPAResponseFileGUID>399D60FDB970C10C3619DC0B378ABF77</EmployerTP
AResponseFileGUID>
    <NumberOfResponseRecordsReceived>1</NumberOfResponseRecordsReceived>
    <NumberOfResponseRecordsInError>0</NumberOfResponseRecordsInError>
    <DateStartedReceivingTransmission>2011-01-21T11:39:45.842-
05:00</DateStartedReceivingTransmission>
    <DateFinishedReceivingTransmission>2011-01-21T11:39:46.310-
05:00</DateFinishedReceivingTransmission>
</EmployerTPAEarningsVerificationResponseCollectionAcknowledgement>
```

> ***Note***: The Broker also sends back custom SOAP header information that tells the overall status of the message. This is defined in Section 4.3- SOAP Custom Headers.

If the Broker determines there were Business Rule or XSD errors in the "Post" message, the Broker will also return to the employer or TPA all of the information that it can on why each individual response failed. The FailedEarningsVerificationResponse element defined in the earnings verification response XSD will present the Error Code and the Error Message of the error it found, as described in Part B, Section C-2.8.

```
<FailedEarningsVerificationResponse>
        <StateEarningsVerificationRequestRecordGUID>4520000000000000000000000
0000001</StateEarningsVerificationRequestRecordGUID>
        <BrokerRecordTransactionNumber>6591</BrokerRecordTransactionNumber>
        <ErrorOccurrence>
            <ErrorCode>452</ErrorCode>
            <ErrorMessage>Business Rule violation - There must be a value
(Numeric - 7.2) for SeveranceAmountPaidForWeek in Repeatable Weekly Earnings
Verification 1 if SeveranceStatusCode (from Request) = 2 for Field Required,
Date Not Required or 3 for Field Required, Date Required, Date Paid or 4 for
Field Required, Date Required, Date Allocated</ErrorMessage>
        </ErrorOccurrence>
    </FailedEarningsVerificationResponse>
```

Putting this together with the successful acknowledgement:

```
<EmployerTPAEarningsVerificationResponseCollectionAcknowledgement
xmlns="https:// REDACTED /schemas">
    <EmployerTPAResponseFileGUID>CD685018CC4E55949E425F86B83E4687</EmployerTP
AResponseFileGUID>
    <FailedEarningsVerificationResponse>
        <StateEarningsVerificationRequestRecordGUID>4520000000000000000000000
0000001</StateEarningsVerificationRequestRecordGUID>
        <BrokerRecordTransactionNumber>6591</BrokerRecordTransactionNumber>
        <ErrorOccurrence>
            <ErrorCode>452</ErrorCode>
            <ErrorMessage>Business Rule violation - There must be a value
(Numeric - 7.2) for SeveranceAmountPaidForWeek in Repeatable Weekly Earnings
Verification 1 if SeveranceStatusCode (from Request) = 2 for Field Required,
Date Not Required or 3 for Field Required, Date Required, Date Paid or 4 for
Field Required, Date Required, Date Allocated</ErrorMessage>
        </ErrorOccurrence>
    </FailedEarningsVerificationResponse>
    <NumberOfResponseRecordsReceived>1</NumberOfResponseRecordsReceived>
    <NumberOfResponseRecordsInError>1</NumberOfResponseRecordsInError>
    <DateStartedReceivingTransmission>2010-12-14T08:18:59.076-
05:00</DateStartedReceivingTransmission>
    <DateFinishedReceivingTransmission>2010-12-14T08:19:07.776-
05:00</DateFinishedReceivingTransmission>
</EmployerTPAEarningsVerificationResponseCollectionAcknowledgement>
```

4.4.2.2 Pull Payload

A "Pull" is defined (see Section 4.1.2) as sending a query to the **Central Broker** to allow the **Central Broker** to send the connector all of the requests or responses it has waiting for it (in multiple transactions if they are from different endpoints). This section discusses the (pre-encryption) payloads in the SOAP message.

4.4.2.2.1 State Pull Payload

4.4.2.2.1.1 Pull from Central Broker

The StateEarningsVerificationResponseCollectionQuery defined in the State Earnings Verification Pull WSDL (see Section 4.6) is a complex type that allows the caller to specify one of three operations: a Pull, a Re-Pull by StateSOAPTransactionNumber, and a Re-Pull by a Date Range.

```
  <!-- Query element for states to collect claim responses they are expecting
-->
    <xs:element name="StateEarningsVerificationResponseCollectionQuery"
        type="StateEarningsVerificationResponseCollectionQueryType"/>

    <!-- Types for query element for states to collect claim responses they
are expecting -->
    <xs:complexType
name="StateEarningsVerificationResponseCollectionQueryType">
        <xs:sequence>
            <xs:element name="StatePostalCode" type="StateAbrCodes" />
            <xs:element
name="StateEarningsVerificationResponseCollectionQueryCriteria"

type="StateEarningsVerificationResponseCollectionQueryCriteriaType"
                minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType
name="StateEarningsVerificationResponseCollectionQueryCriteriaType">
        <xs:sequence>
              <xs:element name="StateSOAPTransactionNumber"
type="xs:nonNegativeInteger" minOccurs="0"/>
              <xs:group
ref="StateEarningsVerificationResponseCollectionQueryCriteriaGroup"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:group
name="StateEarningsVerificationResponseCollectionQueryCriteriaGroup">
        <xs:sequence>
            <xs:element name="BrokerRecordEffectiveDateFrom"
type="CustomDateTime" />
            <xs:element name="BrokerRecordEffectiveDateTo"
type="CustomDateTime" />
        </xs:sequence>

    </xs:group>
```

For the straight Pull, the caller needs to supply only the state Unique ID. Although there are different ways to verify the calling state besides this element, the Broker uses it as an additional security check. Also, there is a requirement in the WSDL 1.1 specification that a WSDL definition have at least one input attribute.

```
<?xml version="1.0" encoding="US-ASCII"?>
<StateEarningsVerificationResponseCollectionQuery xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <StatePostalCode>ST</StatePostalCode>
</StateEarningsVerificationResponseCollectionQuery>
```

For the Re-Pull by StateSOAPTransactionNumber, the caller needs to supply the state Unique ID and the StateSOAPTransactionNumber element out of the StateEarningsVerificationResponseCollectionQueryCriteriaType for Earnings Verification. This will allow the Broker to send the file defined by the StateSOAPTransactionNumber.

```
<?xml version="1.0" encoding="US-ASCII"?>
<StateEarningsVerificationResponseCollectionQuery xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <StatePostalCode>ST</StatePostalCode>
  <StateEarningVerificationResponseCollectionQueryCriteria>
    <StateSOAPTransactionNumber>42153</StateSOAPTransactionNumber>
  </StateEarningsVerificationResponseCollectionQueryCriteria>
</StateEarningsVerificationResponseCollectionQuery>
```

For the Re-Pull by date range, the caller needs to supply the state Unique ID and the StateEarningsVerificationResponseCollectionQueryCriteriaGroup element out of the StateEarningsVerificationResponseCollectionQueryCriteriaType for Earnings Verification. The Re-Pull by date range will pull all the files that were pulled by the connector during the date range specified.

The StateEarningsVerificationResponseCollectionQueryCriteriaGroup is a complex type that is defined as a begin date (BrokerRecordEffectiveDateFrom), an end date (BrokerRecordEffectiveDateTo) and a StateSOAPTransactionNumber.

The first time this operation is called, the StateSOAPTransactionNumber must not be included and the date range that the files to be Re-Pulled are included.

WARNING: If the end date in the Re-Pull by date range is in the future, this will cause the **Central Broker** to resend all transactions ***including*** all the resent transactions that the **Central Broker** has been delivering due to this call, thus putting your Connector into an infinite loop until that date is reached. This will tax the Connector and the **Central Broker** needlessly and must be avoided.

When the Broker sends back the first file in this date range, it will include in the SOAP header the next StateSOAPTransactionNumber that it sent during that date range (in element name NextStateSOAPTransactionNumber). In the next call to this operation, the caller must include the NextStateSOAPTransactionNumber as the StateSOAPTransactionNumber along with the date range. This differentiates to the Broker the next call in the series from a brand new Re-Pull by Date request.

When the **Central Broker** determines that it has no more files to send back to the connector in the given date range, the last file sent back to the connector is indicated by the **Central Broker** not including the next StateSOAPTransactionNumber (so there will not be a NextStateSOAPTransactionNumber included in the http response SOAP header).

First Call:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!-- test query to re-pull by date range, pulls up to 8mb of records -->
<StateEarningsVerificationResponseCollectionQuery xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <StatePostalCode>ST</StatePostalCode>
  <StateEarningsVerificationResponseCollectionQueryCriteria>
    <BrokerRecordEffectiveDateFrom>2009-01-01T00:00:00.000-
04:00</BrokerRecordEffectiveDateFrom>
    <BrokerRecordEffectiveDateTo>2009-12-31T00:00:00.000-
04:00</BrokerRecordEffectiveDateTo>
  </StateEarningsVerificationResponseCollectionQueryCriteria>

</StateEarningsVerificationResponseCollectionQuery>
```

Subsequent Calls:

```
<?xml version="1.0" encoding="US-ASCII"?>
<StateEarningsVerificationResponseCollectionQuery xmlns="https:// REDACTED
/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <StatePostalCode>ST</StatePostalCode>
  <StateEarningsVerificationResponseCollectionQueryCriteria>
  <StateSOAPTransactionNumber>12345678901234567890123456789012<StateSOAP
TransactionNumber>
    <BrokerRecordEffectiveDateFrom>2009-01-01T00:00:00.000-
04:00</BrokerRecordEffectiveDateFrom>
    <BrokerRecordEffectiveDateTo>2009-12-31T00:00:00.000-
04:00</BrokerRecordEffectiveDateTo>
  </StateEarningsVerificationResponseCollectionQueryCriteria>
</StateEarningsVerificationResponseCollectionQuery>
```

4.4.2.2.1.2 Central Broker Response to the State

When the Broker receives a "Pull" request from a State, it begins assembling all the responses that are intended for that state. It constructs a SOAP message according to the rules for a SOAP message (less than 8MB, one employer or TPA per message, etc.). It adds the additional field BrokerRecordEffectiveDate to the response, which indicates the date that it was accepted into the Broker. It then sends the responses in the HTTP response.

```
<StateEarningsVerificationResponseCollection xmlns="https:// REDACTED
/schemas">
    <StateEarningsVerificationResponse>
        <StateEarningsVerificationRequestRecordGUID>AAA5100000000000000000000
```

```
0000001</StateEarningsVerificationRequestRecordGUID>
        <BrokerRecordTransactionNumber>5461</BrokerRecordTransactionNumber>
        <RequestingStateAbbreviation>ST</RequestingStateAbbreviation>
        <UIOfficeName>Office Name</UIOfficeName>
        <StateEmployerAccountNbr>1234567890</StateEmployerAccountNbr>
        <FEIN>123456789</FEIN>
        <CorrectedFEIN>987654321</CorrectedFEIN>
        <EmployerName>ACME</EmployerName>
        <CorrectedEmployerName>Fly By Night</CorrectedEmployerName>
        <SSN>211111111</SSN>
        <ClaimantNameWorkedAsForEmployer>John Q
Public</ClaimantNameWorkedAsForEmployer>
        <NumberofWeeksRequested>5</NumberofWeeksRequested>
        <EarningsVerificationWeekBeginDate>2010-08-
01</EarningsVerificationWeekBeginDate>
        <EarningsVerificationWeekEndDate>2010-09-
04</EarningsVerificationWeekEndDate>
        <ClaimantEmployerWorkRelationshipCode>1</ClaimantEmployerWorkRelation
shipCode>
        <EmployerEarningsCode>1</EmployerEarningsCode>
        <FirstDayWorkedinPeriod>2010-08-01</FirstDayWorkedinPeriod>
        <StillWorkingCode>2</StillWorkingCode>
        <LastDayWorked>2010-09-04</LastDayWorked>
        <EmployerSepReasonCode>1</EmployerSepReasonCode>
        <EarningsVerificationResponseComment>This employee was let go during
the time period</EarningsVerificationResponseComment>
        <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-01</WeekBeginDate>
            <WeekEndDate>2010-08-07</WeekEndDate>
            <HoursWorked>15:00</HoursWorked>
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        </WeeklyEarningsVerification>
        <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-08</WeekBeginDate>
            <WeekEndDate>2010-08-14</WeekEndDate>
            <HoursWorked>15:00</HoursWorked>
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        </WeeklyEarningsVerification>
        <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-15</WeekBeginDate>
            <WeekEndDate>2010-08-21</WeekEndDate>
            <HoursWorked>15:00</HoursWorked>
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        </WeeklyEarningsVerification>
        <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-22</WeekBeginDate>
            <WeekEndDate>2010-08-28</WeekEndDate>
            <HoursWorked>101:00</HoursWorked>
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        </WeeklyEarningsVerification>
        <WeeklyEarningsVerification>
            <WeekBeginDate>2010-08-29</WeekBeginDate>
            <WeekEndDate>2010-09-04</WeekEndDate>
            <HoursWorked>5:00</HoursWorked>
            <AmountEarnedForWeek>500.00</AmountEarnedForWeek>
        </WeeklyEarningsVerification>
        <PreparerTypeCode>T</PreparerTypeCode>
        <PreparerCompanyName>ABC TPA</PreparerCompanyName>
        <PreparerTelephoneNumberPlusExt>5555555556</PreparerTelephoneNumberPl
usExt>
        <PreparerContactName>Mrs Sue Herman</PreparerContactName>
        <PreparerTitle>Claims Administrator</PreparerTitle>
        <PreparerFaxNbr>5555555557</PreparerFaxNbr>
        <PreparerEmailAddress>sue.herman@abctpa.com</PreparerEmailAddress>
        <EarningsVerificationSourceCode>9</EarningsVerificationSourceCode>
        <BrokerRecordEffectiveDate>2011-02-10T13:01:19.000-
05:00</BrokerRecordEffectiveDate>
    </StateEarningsVerificationResponse>
</StateEarningsVerificationResponseCollection>
```

4.4.2.2.1.3 State Acknowledgement to the Central Broker

The StateEarningsVerificationResponseCollectionAcknowledgement is initiated once the state has received its file from the Broker. The acknowledgement must accompany every state Pull request, as this is the manner in which the Broker knows that the state Pull was successful. This is required even if the Broker has sent back an empty file and a MessageCode of 2. If this is not sent back to the Broker, the next “Pull” call to the Broker will result in the same file being passed back. The Broker will not move on to the next file until it receives a successful acknowledgement. If the Broker receives 3 unanswered Pull requests, it will suspend any processing of Pull requests by the State until the **Broker Administrator** and State Administrator can work out the problem.

The key field in this message is the StateSOAPTransmissionNumber, which must correspond with the StateSOAPTransmissionNumber sent back in the Broker Response. The remainder of the message is just reporting information; the values are not used for anything at this time. If the state does not collect this information, just return 0 for the number of records and place a valid date in the date fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEarningsVerificationResponseCollectionAcknowledgement xmlns="https://
REDACTED /schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED / REDACTED ">

<StateSOAPTransmissionNumber>12345678901234567890123456789012</StateSOAPTrans
missionNumber>
  <NumberOfResponseRecordsReceived>0</NumberOfResponseRecordsReceived>
  <NumberOfResponseRecordsInError>0</NumberOfResponseRecordsInError>
  <DateStartedReceivingTransmission>2001-12-31T12:00:00.00-
04:00</DateStartedReceivingTransmission>
  <DateFinishedReceivingTransmission>2001-12-31T12:00:00.00-
04:00</DateFinishedReceivingTransmission>
</StateEarningsVerificationResponseCollectionAcknowledgement>
```

4.4.2.2.2 Employer/TPA Pull

4.4.2.2.2.1 Pull from Central Broker

The EmployerTPAEarningsVerificationRequestCollectionQuery is a complex query types that allow the caller to specify one of three operations: a Pull, a Re-Pull by EmployerTPASOAPTransactionNumber, and a Re-Pull by a Date Range.

```
<!-- Query element for employer to collect claim responses they are expecting
-->
    <xs:element name="EmployerTPAEarningsVerificationRequestCollectionQuery"
        type="EmployerTPAEarningsVerificationRequestCollectionQueryType"/>

    <!-- Types for query element for Employers/TPAs to collect claim requests
they are expecting -->
    <xs:complexType
name="EmployerTPAEarningsVerificationRequestCollectionQueryType">
        <xs:sequence>
            <xs:element name="UniqueID" type="UniqueIDType" />
            <xs:element
name="EmployerTPAEarningsVerificationRequestCollectionQueryCriteria"

type="EmployerTPAEarningsVerificationRequestCollectionQueryCriteriaType"
                minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType
name="EmployerTPAEarningsVerificationRequestCollectionQueryCriteriaType">
        <xs:sequence>
              <xs:element name="EmployerTPASOAPTransactionNumber"
type="xs:nonNegativeInteger" minOccurs="0" />
            <xs:group
ref="EmployerTPAEarningsVerificationRequestCollectionQueryCriteriaGroup"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:group
name="EmployerTPAEarningsVerificationRequestCollectionQueryCriteriaGroup">
        <xs:sequence>
            <xs:element name="BrokerRecordEffectiveDateFrom"
type="CustomDateTime" />
            <xs:element name="BrokerRecordEffectiveDateTo"
type="CustomDateTime" />
        </xs:sequence>
    </xs:group>
```

For the straight Pull, the caller needs to supply only the employer or TPA Unique ID. Although there are different ways to verify the calling employer or TPA besides this element, the Broker uses it as an additional security check. Also, there is a requirement in WSDL 1.1 that a WSDL definition have at least one input attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<EmployerTPAEarningsVerificationRequestCollectionQuery xmlns="https://
REDACTED /schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <UniqueID>BR999999999</UniqueID>
</EmployerTPAEarningsVerificationRequestCollectionQuery>
```

For the Re-Pull by EmployerTPASOAPTransactionNumber, the caller needs to supply the Employer/TPA Unique ID and the EmployerTPASOAPTransactionNumber element out of the EmployerTPAEarningsVerificationResponseCollectionQueryCriteriaType. This will allow the Broker to send the file defined by the EmployerTPASOAPTransactionNumber.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPAEarningsVerificationRequestCollectionQuery xmlns="https://
REDACTED /schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <UniqueID>BR999999999</UniqueID>
  <EmployerTPAEarningsVerificationRequestCollectionQueryCriteria>
      <EmployerTPASOAPTransactionNumber>45459</EmployerTPASOAPTransactionNumb
      er>
  </EmployerTPAEarningsVerificationRequestCollectionQueryCriteria>
</EmployerTPAEarningsVerificationRequestCollectionQuery>
```

For the Re-Pull by date range, the caller needs to supply the EmployerTPA Unique ID and the EmployerTPAEarningsVerificationResponseCollectionQueryCriteriaGroup element out of the EmployerTPAEarningsVerificationResponseCollectionQueryCriteriaType. The re-pull by date range will pull all the files that were previously pulled by the connector during the date range specified.

The EmployerTPAEarningsVerificationResponseCollectionQueryCriteriaGroup is a complex query type that is defined as a begin date (BrokerRecordEffectiveDateFrom), an end date (BrokerRecordEffectiveDateTo) and an EmployerTPASOAPTransactionNumber.

WARNING: If the end date in the Re-Pull by date range is in the future, this will cause the **Central Broker** to resend all transactions ***including*** all the resent transactions that the **Central Broker** has been delivering due to this call, thus putting your Connector into an infinite loop until that date is reached. This will tax the Connector and the **Central Broker** needlessly and must be avoided.

The first time this operation is called, the EmployerTPASOAPTransactionNumber must not be included and the date range that the files to be Re-Pulled are included.

When the Broker sends back the first file in this date range, it will include in the SOAP header the next EmployerTPASOAPTransactionNumber that it sent during that date range (in element name NextEmployerTPASOAPTransactionNumber). In the next call to this operation, the caller must include the NextEmployerTPASOAPTransactionNumber as the

EmployerTPASOAPTransactionNumber along with the date range. This differentiates to the Broker the next call in the series from a brand new Re-Pull by Date request.

When the **Central Broker** determines that it has no more files to send back to the connector in the given date range, the last file sent back to the connector is indicated by the **Central Broker** not including the next EmployerTPASOAPTransactionNumber (so there will not be a NextEmployerTPASOAPTransactionNumber included in the http response SOAP header).

First Call:

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPAEarningsVerificationRequestCollectionQuery xmlns="https://
REDACTED /schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <UniqueID>BR999999999</UniqueID>
  <EmployerTPAEarningsVerificationRequestCollectionQueryCriteria>
    <BrokerRecordEffectiveDateFrom>2009-01-01T00:00:00.000-
04:00</BrokerRecordEffectiveDateFrom>
    <BrokerRecordEffectiveDateTo>2010-12-31T00:00:00.000-
04:00</BrokerRecordEffectiveDateTo>
  </EmployerTPAEarningsVerificationRequestCollectionQueryCriteria>
</EmployerTPAEarningsVerificationRequestCollectionQuery>
```

Subsequent Calls:

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPAEarningsVerificationRequestCollectionQuery xmlns="https://
REDACTED /schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <UniqueID>BR999999999</UniqueID>
  <EmployerTPAEarningsVerificationRequestCollectionQueryCriteria>

<EmployerTPASOAPTransactionNumber>26101</EmployerTPASOAPTransactionNumber>
    <BrokerRecordEffectiveDateFrom>2009-01-01T00:00:00.000-
04:00</BrokerRecordEffectiveDateFrom>
    <BrokerRecordEffectiveDateTo>2010-12-31T00:00:00.000-
04:00</BrokerRecordEffectiveDateTo>
  </EmployerTPAEarningsVerificationRequestCollectionQueryCriteria>
</EmployerTPAEarningsVerificationRequestCollectionQuery>
```

4.4.2.2.2.2 Central Broker Response to Employer/TPA

When the Broker receives a “Pull” request from an Employer/TPA, it begins assembling all the earnings verification requests that are intended for that employer or TPA. It constructs a SOAP message according to the rules for a SOAP message (less than 8MB, one employer or TPA per message, etc.). It adds two additional fields to the Request - the BrokerRecordEffectiveDate and the BrokerRecordTransactionNumber. The BrokerRecordEffectiveDate indicates the date that it was accepted into the Broker. The BrokerRecordTransactionNumber is a unique record tracking number and must be returned on the response for this record. It then sends the requests in the HTTP response.

```
<EmployerTPAEarningsVerificationRequestCollection xmlns="https:// REDACTED
/schemas">
    <EmployerTPAEarningsVerificationRequest>
        <StateEarningsVerificationRequestRecordGUID>2015000000000000000000000
0000001</StateEarningsVerificationRequestRecordGUID>
        <RequestingStateAbbreviation>ST</RequestingStateAbbreviation>
        <UIOfficeName>Office Name</UIOfficeName>
        <UIOfficePhone>5555555555</UIOfficePhone>
        <UIOfficeFax>5555555554</UIOfficeFax>
        <UIOfficeEmailAddress>james.madison@state.gov</UIOfficeEmailAddress>
        <StateEmployerAccountNbr>1234567890</StateEmployerAccountNbr>
        <FEIN>123456789</FEIN>
        <EmployerName>ACME</EmployerName>
        <SSN>213456721</SSN>
        <ClaimantLastName>Lastname</ClaimantLastName>
        <ClaimantFirstName>Firstname</ClaimantFirstName>
        <ClaimantMiddleInitial>M</ClaimantMiddleInitial>
        <ClaimantSuffix>JR</ClaimantSuffix>
        <NumberofWeeksRequested>5</NumberofWeeksRequested>
        <EarningsVerificationWeekBeginDate>2010-08-
01</EarningsVerificationWeekBeginDate>
        <EarningsVerificationWeekEndDate>2010-09-
04</EarningsVerificationWeekEndDate>
        <EarningsVerificationComments>This is a comment field for this
Earnings Verification Request</EarningsVerificationComments>
        <RequestDate>2010-10-14</RequestDate>
        <EarningsStatusCode>3</EarningsStatusCode>
        <TipsStatusCode>3</TipsStatusCode>
        <CommissionStatusCode>3</CommissionStatusCode>
        <BonusStatusCode>3</BonusStatusCode>
        <VacationStatusCode>3</VacationStatusCode>
        <SickLeaveStatusCode>3</SickLeaveStatusCode>
        <HolidayStatusCode>3</HolidayStatusCode>
        <SeveranceStatusCode>3</SeveranceStatusCode>
        <WagesInLieuStatusCode>3</WagesInLieuStatusCode>
        <EarningsVerificationResponseCommentIndicator>1</EarningsVerification
ResponseCommentIndicator>
        <ResponseDueDate>2010-10-28</ResponseDueDate>
        <EarningsVerificationSourceCode>9</EarningsVerificationSourceCode>
        <BrokerRecordTransactionNumber>6609</BrokerRecordTransactionNumber>
        <BrokerRecordEffectiveDate>2011-03-11T13:09:20.000-
05:00</BrokerRecordEffectiveDate>
    </EmployerTPAEarningsVerificationRequest>
</EmployerTPAEarningsVerificationRequestCollection>
```

4.4.2.2.2.3 Employer/TPA Acknowledgement to Central Broker

The EmployerTPAEarningsVerificationRequestCollectionAcknowledgement is initiated once the employer or TPA has received its file from the Broker. The acknowledgement must accompany every employer or TPA Pull request, as this is the manner in which the Broker knows that the Employer/TPA Pull was successful. This is required even if the Broker has sent back an empty file and a MessageCode of 2. If this is not sent back to the Broker, the next

"Pull" call to the Broker will result in the same file being passed back. The Broker will not move on to the next file until it receives a successful acknowledgement. If the Broker receives 3 unanswered Pull requests, it will suspend any processing of Pull requests by the Employer/TPA until the **Broker Administrator** and Employer/TPA Administrator can work out the problem.

The key field in this message is the EmployerTPASOAPTransmissionNumber, which must correspond with the EmployerTPASOAPTransmissionNumber sent back in the Broker Response. The remainder of the message is just reporting information; the values are not used for anything at this time. If the EmployerTPA does not collect this information, just return 0 for the number of records and place a valid date in the date fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmployerTPAEarningsVerificationRequestCollectionAcknowledgement
xmlns="https:// REDACTED /schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https:// REDACTED /schemas REDACTED ">
  <EmployerTPASOAPTransmissionNumber>3400</EmployerTPASOAPTransmissionNumber>
  <NumberOfRequestRecordsReceived>4</NumberOfRequestRecordsReceived>
  <NumberOfRequestRecordsInError>0</NumberOfRequestRecordsInError>
  <DateStartedReceivingTransmission>2001-12-31T12:00:00.000-
04:00</DateStartedReceivingTransmission>
  <DateFinishedReceivingTransmission>2001-12-31T12:00:00.000-
04:00</DateFinishedReceivingTransmission>
</EmployerTPAEarningsVerificationRequestCollectionAcknowledgement>
```

4.5 SOAP Action

From the W3C, the SOAPAction component in SOAP 1.1 is defined as follows:

"The SOAPAction HTTP request header field can be used to indicate the intent of the SOAP HTTP request. The value is a URI identifying the intent. SOAP places no restrictions on the format or specificity of the URI or that it is resolvable. An HTTP client MUST use this header field when issuing a SOAP HTTP Request."

The SOAPAction **MUST** be specified in the SOAP message. The particular SOAP Action required for each message is specified in the WSDL.

There are six SOAP Actions that are defined in **SIDES** for Separation Information:

1. postStateSeparationRequestCollection

2. pullStateSeparationResponseCollection

3. pullStateSeparationResponseCollectionAcknowledgement

4. postEmployerTPASeparationResponseCollection
5. pullEmployerTPASeparationRequestCollection
6. pullEmployerTPASeparationRequestCollectionAcknowledgement

There are six SOAP Actions that are defined in **SIDES** for Earnings Verification:

1. postStateEarningsVerificationRequestCollection
2. pullStateEarningsVerificationResponseCollection
3. pullStateEarningsVerificationResponseCollectionAcknowledgement
4. postEmployerTPAEarningsVerificationResponseCollection
5. pullEmployerTPAEarningsVerificationRequestCollection
6. pullEmployerTPAEarningsVerificationRequestCollectionAcknowledgement

4.6 WSDL

The Web Services Description Language (WSDL) is an XML-based language that provides a model for describing Web services. WSDL defines an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

The WSDL for **SIDES** is broken into two files per Standard Format. StateBroker.wsdl for Separation Information and EarningsVerificationStateBroker.wsdl for Earnings Verification describe the interfaces exposed to the States from the Broker. EmployerTPABroker.wsdl for Separation Information and EarningsVerificationEmployerTPABroker.wsdl for Earnings Verification describe the interfaces exposed to the employer or /TPA from the Broker. Note that the Push to the employer or TPA is not described in these WSDLs because that function belongs in the WSDL for the connector Web Service. (See the full WSDL below for a complete description.)

- State Separation Information WSDL:

  REDACTED

- Employer/TPA Separation Information WSDL:

  REDACTED

- State Earnings Verification WSDL:

  REDACTED

- Employer/TPA Earnings Verification WSDL:

  REDACTED

There is no Policy information contained within these WSDLs. This was done for interoperability reasons. If a connector is using a technology that requires the use of Policy information, a state WSDL has been provided within the JAX-WS Model Connector.

4.6.1 WSDL XSD

Along with each WSDL, there are XSD files that define the elements in the WSDL. These schema files are StateTransmissionQuery.xsd and EmployerTPATransmissionQuery.xsd for Separation Information, and EarningsVerificationStateTransmissionQuery.xsd and EarningsVerificationEmployerTPATransmissionQuery.xsd for Earnings Verification. There is one support file - TransmissionQueryCommonElements.xsd – used by both exchanges. The XSDs for the current data elements can be found at:

- Separation Information

  REDACTED

- Earnings Verification

  REDACTED

- Common Elements

  REDACTED

One XSD file, combined.xsd, is used to include other XSD files in the system and it does not contain any additional information. This file is required due to a problem accessing the https:// **REDACTED** /schemas namespace in multiple files within the Java libraries used in **SIDES**. The combined.xsd file is used internally by the **Central Broker** to allow XSD checks to take place on all the SOAP messages and records sent in by the connectors.

The combined.xsd file may be used by connector software, but it is not necessary if the technology and libraries used in the connectors' implementation do not require it. The combined.xsd file can be found at:

https:// **REDACTED** /schemas/combined.xsd

4.6.2 State WSDL

4.6.2.1 State Post WSDL

4.6.2.1.1 Separation Information State Post WSDL

The WSDL for the state Post operation in Separation Information is defined as

```
<wsdl:operation name="postStateSeparationRequestCollection">
      <soap:operation soapAction="postStateSeparationRequestCollection" />
      <wsdl:input name="StateSeparationRequestCollection">
```

```
            <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output name="StateSeparationRequestCollectionAcknowledgement">
            <soap:body use="literal" />
      </wsdl:output>
</wsdl:operation>
```

The input (HTTP request) to this operation is a StateSeparationRequestCollection as defined in SeparationRequest.xsd.

The output (HTTP response) is a StateSeparationRequestCollectionAcknowledgement as defined in the SeparationRequest.xsd.

4.6.2.1.2 Earnings Verification State Post WSDL

The WSDL for the state Post operation in Earnings Verification is defined as

```
<wsdl:operation name="postStateEarningsVerificationRequestCollection">
      <wsdl:input message="tns:StateEarningsVerificationRequestCollection"
                  name="StateEarningsVerificationRequestCollection">
      </wsdl:input>
      <wsdl:output
message="tns:StateEarningsVerificationRequestCollectionAcknowledgement"
            name="StateEarningsVerificationRequestCollectionAcknowledgement">
      </wsdl:output>
</wsdl:operation>
```

The input (HTTP request) to this operation is a StateEarningsVerificationRequestCollection as defined in EarningsVerificationRequest.xsd.

The output (HTTP response) is a StateEarningsVerificationRequestCollectionAcknowledgement as defined in the EarningsVerificationRequest.xsd.

4.6.2.2 State Pull WSDL

4.6.2.2.1 Separation Information State Pull WSDL

The WSDL for the state Pull operation in Separation Information is defined as:

```
<wsdl:operation name="pullStateSeparationResponseCollection">
      <soap:operation soapAction="pullStateSeparationResponseCollection" />
      <wsdl:input name="StateSeparationResponseCollectionQuery">
            <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output name="StateSeparationResponseCollection">
            <soap:body use="literal" />
      </wsdl:output>
</wsdl:operation>
```

The input (HTTP request) to this operation is a StateSeparationResponseCollectionQuery as defined in StateTransmissionQuery.xsd.

The output (HTTP response) is a StateSeparationResponseCollection as defined in the SeparationResponse.xsd.

4.6.2.2.2 Earnings Verification State Pull WSDL

The WSDL for the state Pull operation in Earnings Verification is defined as:

```
<wsdl:operation name="pullStateEarningsVerificationResponseCollection">
      <wsdl:input
message="tns:StateEarningsVerificationResponseCollectionQuery"
           name="StateEarningsVerificationResponseCollectionQuery">
      </wsdl:input>
      <wsdl:output message="tns:StateEarningsVerificationResponseCollection"
           name="StateEarningsVerificationResponseCollection">
      </wsdl:output>
</wsdl:operation>
```

The input (HTTP request) to this operation is a StateEarningsVerificationResponseCollectionQuery as defined in EarningsVerificationStateTransmissionQuery.xsd.

The output (HTTP response) is a StateEarningsVerificationResponseCollection as defined in the EarningsVerificationResponse.xsd.

4.6.3 Employer/TPA WSDL

4.6.3.1 EmployerTPA Post WSDL

4.6.3.1.1 Separation Information EmployerTPA Post WSDL

The WSDL for the employer or TPA Post operation in Separation Information is defined as:

```
<wsdl:operation name="postEmployerTPASeparationResponseCollection">
      <soap:operation
            soapAction="postEmployerTPASeparationResponseCollection" />
      <wsdl:input name="EmployerTPASeparationResponseCollection">
            <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output
            name="EmployerTPASeparationResponseCollectionAcknowledgement">
            <soap:body use="literal" />
      </wsdl:output>
</wsdl:operation>
```

The input (HTTP request) to this operation is an EmployerTPASeparationResponseCollection as defined in SeparationResponse.xsd.

The output (HTTP response) is an EmployerTPASeparationResponseCollectionAcknowledgement as defined in the SeparationResponse.xsd.

4.6.3.1.2 Earnings Verification EmployerTPA Post WSDL

The WSDL for the employer or TPA Post operation in Earnings Verification is defined as:

```
<wsdl:operation name="postEmployerTPAEarningsVerificationResponseCollection">
  <wsdl:input message="tns:EmployerTPAEarningsVerificationResponseCollection"
     name="EmployerTPAEarningsVerificationResponseCollection">
  </wsdl:input>
  <wsdl:output
message="tns:EmployerTPAEarningsVerificationResponseCollectionAcknowledgement
"
name="EmployerTPAEarningsVerificationResponseCollectionAcknowledgement">
  </wsdl:output>

</wsdl:operation>
```

The input (HTTP request) to this operation is an EmployerTPAEarningsVerificationResponseCollection as defined in EarningsVerificationResponse.xsd.

The output (HTTP response) is an EmployerTPAEarningsVerificationResponseCollectionAcknowledgement as defined in the EarningsVerificationResponse.xsd.

4.6.3.2 EmployerTPA Pull WSDL

4.6.3.2.1 Separation Information EmployerTPA Pull WSDL

The EmployerTPA Pull operation for Separation Information is defined as

```
<wsdl:operation name="pullEmployerTPASeparationRequestCollection">
   <soap:operation soapAction="pullEmployerTPASeparationRequestCollection" />
   <wsdl:input name="EmployerTPASeparationRequestCollectionQuery">
         <soap:body use="literal" />
   </wsdl:input>
   <wsdl:output name="EmployerTPASeparationRequestCollection">
         <soap:body use="literal" />
   </wsdl:output>
</wsdl:operation>
```

The input (HTTP request) to this operation is an EmployerTPASeparationRequestCollectionQuery as defined in EmployerTPATransmissionQuery.xsd.

The output (HTTP response) is an EmployerTPASeparationRequestCollection as defined in the SeparationRequest.xsd.

4.6.3.2.2 Earnings Verification EmployerTPA Pull WSDL

The EmployerTPA Pull operation for Earnings Verification is defined as

```
<wsdl:operation name="pullEmployerTPAEarningsVerificationRequestCollection">
  <wsdl:input
message="tns:EmployerTPAEarningsVerificationRequestCollectionQuery"
    name="EmployerTPAEarningsVerificationRequestCollectionQuery">
  </wsdl:input>
  <wsdl:output message="tns:EmployerTPAEarningsVerificationRequestCollection"
    name="EmployerTPAEarningsVerificationRequestCollection">
  </wsdl:output>
</wsdl:operation>
```

The input (HTTP request) to this operation is an EmployerTPAEarningsVerificationRequestCollectionQuery as defined in EarningsVerificationEmployerTPATransmissionQuery.xsd.

The output (HTTP response) is an EmployerTPAEarningsVerificationRequestCollection as defined in the EarningsVerificationRequest.xsd.

5 C – BUILD THE CONNECTOR: SECURING THE MESSAGE

REDACTED

6 D - CONNECT WITH THE CENTRAL BROKER: SENDING THE MESSAGE

6.1 Sending a message

The destination address for all messaging with the **Central Broker** for the Separation Information exchange is:

REDACTED

The destination address for all messaging with the **Central Broker** for the Earnings Verification exchange is:

REDACTED

6.2 Sample SOAP message sent

Below is a sample of a State SOAP message that a connector has sent to the Broker for Separation Information:

Sample State SOAP Message

REDACTED

6.3 Acknowledgements

The acknowledgement in any of the message types is an important part of the transaction. They let the receiver know that the file has successfully made it to its destination, regardless of the individual records success or failure. If an acknowledgement is not received within 15 minutes of sending a message, failure must be assumed and the message should be sent again.

A message failure and re-send could be an infinite cycle if one of the connectors is having difficulty. Therefore, connectors must be implemented to limit this process to occur only three times (an initial time plus two retries) in an automated fashion before contacting the connector's **SIDES** administrator. The state, employer, or TPA administrator can then begin debugging the problem. If the problem was noted by the **Central Broker**, the **SIDES** **Broker Administrator** will be notified and error resolution will be started on the **Central Broker** side also.

6.4 Non-Broker Returns

There are some messages that a state, employer, or TPA connector may receive when attempting to communicate with the **Central Broker** that are non-standard **Central Broker** return messages. These messages were not discussed earlier in this document as part of the message exchange with the **Central Broker** as they are not message related but, rather, are overall system-related return messages. These must be handled by all **Central Broker** client connectors.

1. “UI **SIDES** Server Error has occurred. Please try again in a few minutes or contact your UI **SIDES** Administrator for assistance.”

   REDACTED

2. A no response. A [404] Http Error.

   REDACTED

7 E – CONNECT WITH THE CENTRAL BROKER: TESTING CONNECTOR SOFTWARE

The connector software must be designed, coded and thoroughly tested to ensure correct functionality when interfacing with the **Central Broker**. Connector testing responsibilities are highlighted as well as key points for consideration when testing your connector software. To support connector testing, a set of tools are available to aid in the development and testing process.

7.1 Connector Responsibility

It is the state or employer/TPA's responsibility to fully test their **SIDES** implementation. Connector testing includes the backend system components, system interfaces, and the connector software that interacts with the **Central Broker**. The connector's process for testing is not prescribed as different organizations must follow their own testing procedures and standards. However, to test the connector with the **Central Broker**, the **SIDES** certification test process (see Section 8) including injection of XML certification test data (see section 9.2.2.1) must be followed.

This Developer Guide not only provides developers with a roadmap for constructing the connector, but also serves as a tool the test team may use to support comprehensive testing of the connector software. The following list highlights key points to be considered when testing your connector software:

- Creation of the Request or Response XML
    - All the data required in the Standard Format is being accessed correctly
- The XML data format (Standard Format)
    - XSD violations of the Standard Format must be trapped and handled on the connector side.
    - For a State using the Separation Information Standard Format, all the ClaimantSepReasonCodes have been tested along with any business rule dependencies on the individual ClaimantSepReasonCode.
    - For the employer/TPAs using the Separation Information Standard Format, all the EmployerSepReasonCodes have been tested along with any business rule dependencies on the individual EmployerSepReasonCode.
    - Other values that are called out in the Standard Format. A test case should not only be made for the negative (error) conditions, but also the positive (non error) conditions – especially around the boundaries.
    - The entire set of error codes, which result from business rule violations, must be handled. This must be performed to cover the case where an error code is received

    internally before the message is generated and after the message is sent and the error code comes back in the acknowledgment.

- SOAP concepts
    - All of the required custom headers are included with the message.
    - The SOAP action is part of the https message.
    - Generation of the digital signature and encryption occurs correctly.
    - The message timestamp has a 15 minute time to live.
- Message Codes
    - **Central Broker** generated message codes on the acknowledgment of a Post and an http response on the Pull.
    - Connector generated message codes on the acknowledgement of a Pull.
- Duplicate records
    - Duplicate records Pulled in the same file and in a different file.

7.2 Tools

This section describes the tools available to aid development and testing of the connector software. The first tool provided is a set of state and employer/TPA Model Connectors. The Model Connectors serve two main purposes: 1) to provide a road map in creating the connector software and 2) to provide the connector with an “opposing” endpoint they can send/receive messages to and from.

The second tool provided is a business rule processor tool (BRPT). The BRPT also has two uses. The first use is to provide a simulated **Central Broker**, allowing a connector to simulate exchanging messages with the **Central Broker** while their connection piece is being constructed, prior to connection the real **Central Broker**. The other use of the BRPT is to verify the connector software business rules are behaving in the same manner as the **Central Broker**.

7.2.1 Model Connectors

The state and employer/TPA Model Connectors are a set of tools that allow a proven method of communicating with the **SIDES** Broker right out of the box. These components may be used as a “black-box”, which may be linked with the Endpoint’s back-end software to facilitate integration with the Central Broker. Alternatively, **SIDES** connector developers may construct their own **SIDES** connector, and use the Model Connector to help with all aspects to connect with the Central Broker including security and the proper construction of the SOAP message.

The Model Connectors are available in the following technologies:

- Spring-WS - Java
- JAX-WS – Java
- Microsoft .Net – C#

The Model Connectors are designed to operate as a “black-box” so states, employers, and TPAs do not have to build their own connector software. Please be aware that the Model Connectors will be updated to accommodate new SIDES exchanges as they are completed. Also, periodic updates to the Model Connectors may be released to address software enhancements or to remediate defects. Source code for the Model Connectors are provided so states, employers, or TPAs may make adjustments (if necessary) to operate within their environment. Please be aware that the SIDES Team will not carry forward any custom software changes made to the Model Connectors, and the Endpoints must re-apply the changes to new versions of the Model Connector.

Prior to deployment of the Model Connector into production operations, the state, employer, or TPA, must conduct testing with real scenarios/data against the SIDES Broker Test environment. This testing is imperative to ensure:

- Proper integration with the back-end system
- Correct processing and interpretation of log files, request files, and response files
- There are no unanticipated data sets, which may not be processed by the Model Connector.

Contact the Broker Team to discuss any Model Connector enhancements or to report software defects.

The Model Connectors can also provide an opposite endpoint (employer/TPA for states; states for employer/TPAs) to help with the developer’s end-to-end testing without the need for an actual endpoint to be participating.

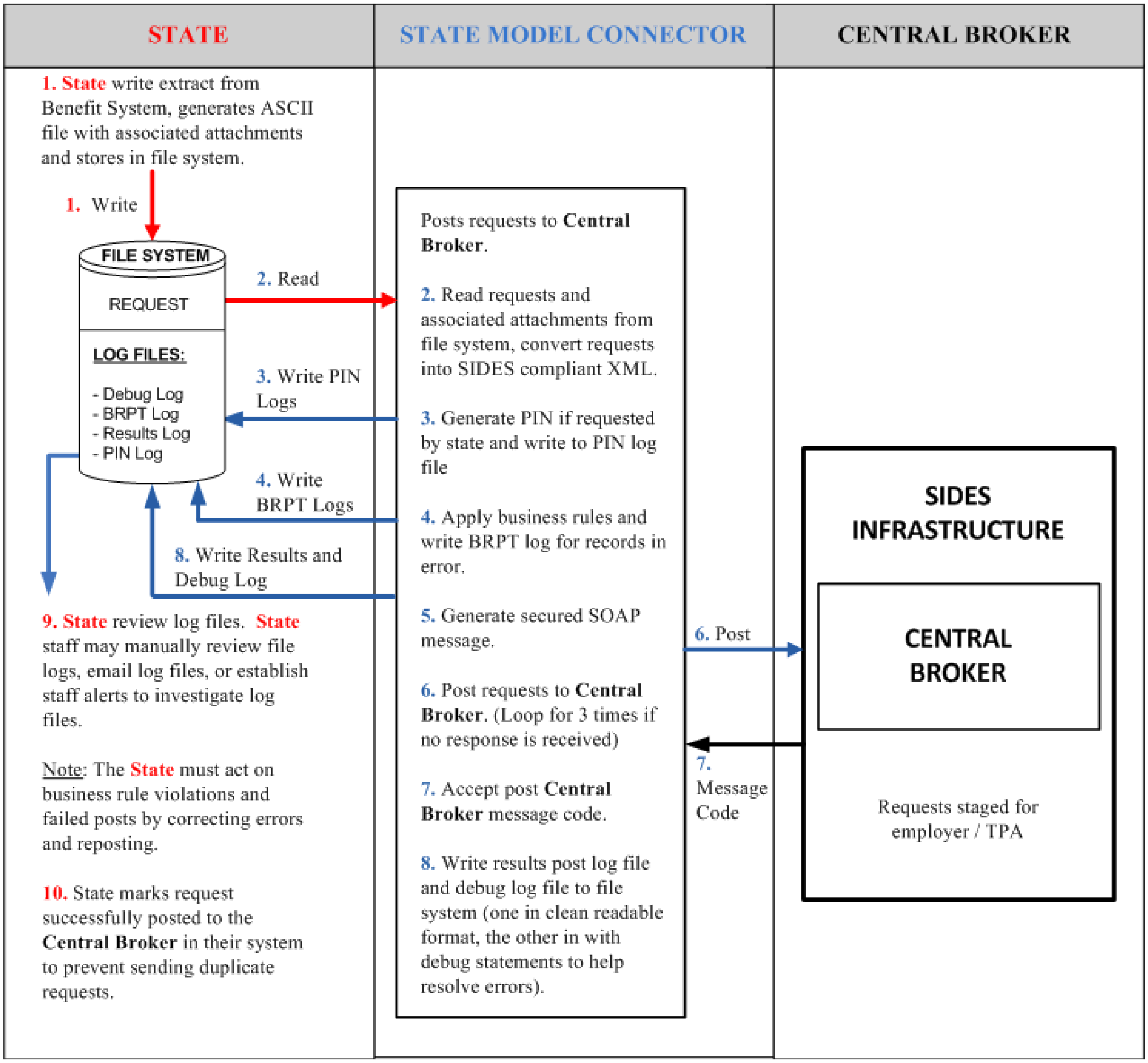
As an example, consider where ‘State X’ is in initial construction of its software. ‘State X’ needs to produce messages and consume messages through the **Central Broker**. ‘State X’ wants to know what the messages that they send to the employer/TPA look like, and they need an employer/TPA to answer their requests in order to test their consumption portion. The emulated employer/TPA can act as the endpoint that the State is communicating without relying on an actual employer/TPA system on the other side of the Broker.

‘State X’ can accomplish this simulation in the following manner.

1. ‘State X’ creates request files to be sent to their emulated employer/TPA, ‘employer/TPA Y’. Note: The emulated ‘employer/TPA Y’ has already been created in the **Central Broker** by the **Broker Administrator**.
2. ‘State X’ Posts the file to ‘employer/TPA Y’.
3. ‘State X’, using the employer/TPA Model Connector, Pulls messages from the **Central Broker** as ‘employer/TPA Y’. This gives ‘State X’ the BrokerRecordTransactionNumbers of all of the requests.
4. ‘State X’ creates responses to their own requests, filling in the backfilled data as required by employer/TPAs.
5. Using the employer/TPA Model Connector, ‘State X’ Posts all of ‘employer/TPA Y’’s responses back to themselves.
6. ‘State X’ can then Pull responses from the **Central Broker** which will include ‘employer/TPA Y’’s responses.

For the Pull action, an acknowledgement is automatically sent to the Broker after a Pull action downloads the file from the Broker. This ensures the Model Connector completes the full Pull action.

The Model Connector demonstrates how a state can access the UI **SIDES** Broker Web services using the different Model Connector libraries. It can accomplish both a Post to the **Central Broker** or a Pull from the **Central Broker**. There are two ways it can accomplish these actions. The first method is by calling the Model Connector with the XML payload and the SOAP header values as input. The second method is by calling the Model Connector with an ASCII data file that contains the same data as within the XML file and the SOAP Headers. Figure 1 and Figure 2 are detailed diagrams of steps taken by the State, the Model Connector and the **Central Broker** for both the Post and Pull with the ASCII file. The ASCII file is a new format and a discussion of this format is below.



1. State write extract from Benefit System, generates ASCII file with associated attachments and stores in file system.

9. State review log files. State staff may manually review file logs, email log files, or establish staff alerts to investigate log files.

Note: The State must act on business rule violations and failed posts by correcting errors and reposting.

10. State marks request successfully posted to the **Central Broker** in their system to prevent sending duplicate requests.

Posts requests to **Central Broker**.

2. Read requests and associated attachments from file system, convert requests into SIDES compliant XML.

3. Generate PIN if requested by state and write to PIN log file

4. Apply business rules and write BRPT log for records in error.

5. Generate secured SOAP message.

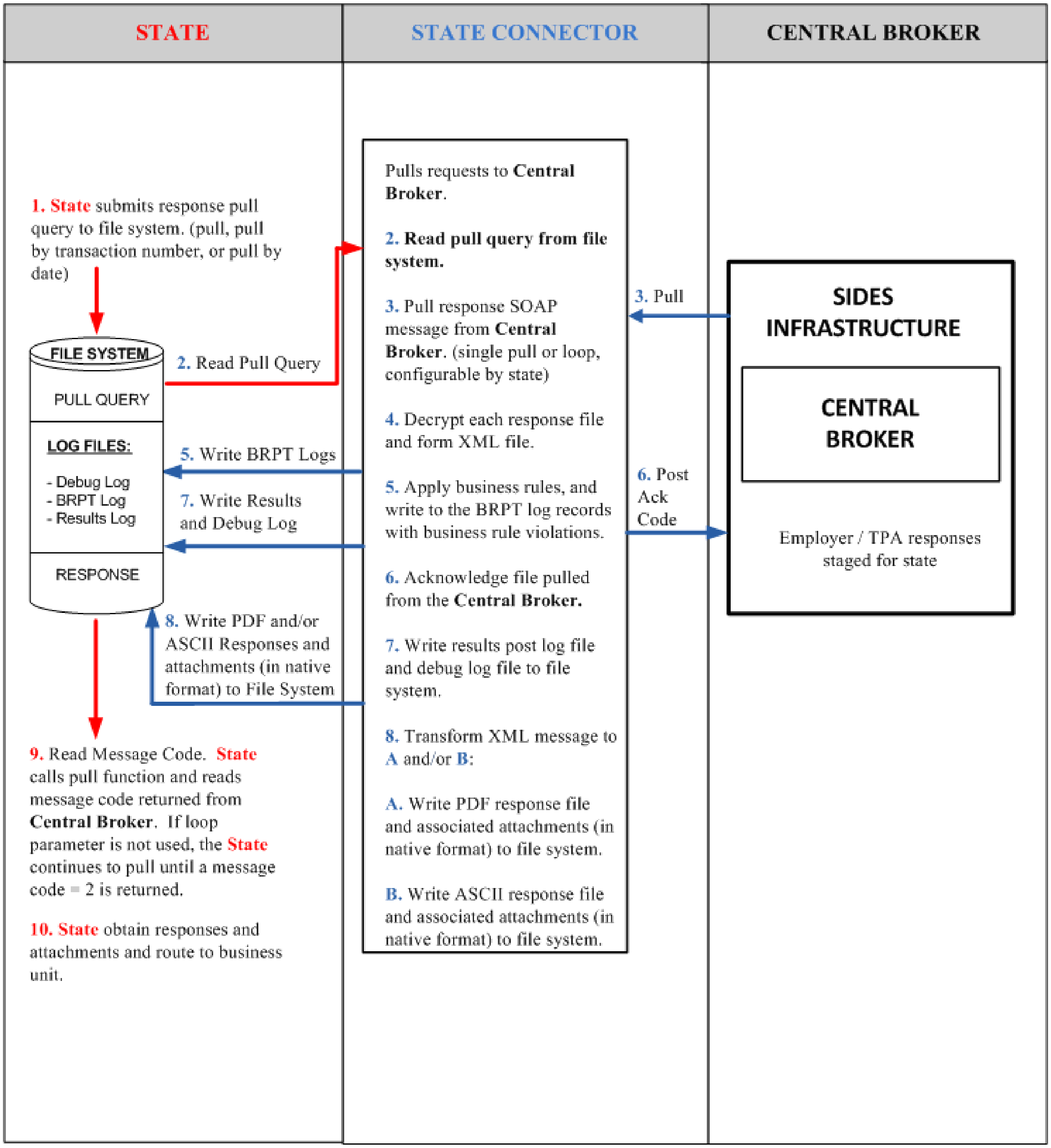
6. Post requests to **Central Broker**. (Loop for 3 times if no response is received)

7. Accept post **Central Broker** message code.

8. Write results post log file and debug log file to file system (one in clean readable format, the other in with debug statements to help resolve errors).

Requests staged for employer / TPA

Figure 1. Model Connector Post for ASCII Files



1. State submits response pull query to file system. (pull, pull by transaction number, or pull by date)

8. Write PDF and/or ASCII Responses and attachments (in native format) to File System

9. Read Message Code. State calls pull function and reads message code returned from **Central Broker**. If loop parameter is not used, the State continues to pull until a message code = 2 is returned.

10. State obtain responses and attachments and route to business unit.

Pulls requests to **Central Broker**.

2. **Read pull query from file system.**

3. Pull response SOAP message from **Central Broker**. (single pull or loop, configurable by state)

4. Decrypt each response file and form XML file.

5. Apply business rules, and write to the BRPT log records with business rule violations.

6. Acknowledge file pulled from the **Central Broker.**

7. Write results post log file and debug log file to file system.

8. Transform XML message to A and/or B:

A. Write PDF response file and associated attachments (in native format) to file system.

B. Write ASCII response file and associated attachments (in native format) to file system.

Figure 2. Model Connector Pull for ASCII Files

7.2.1.1 Setup State Model Connector

To begin using the State Model Connector, the state must first download the software from the sides.itsc.org website.

There are two options for download of the State Model Connector.

The first option is the ‘Black Box’ approach. This download contains only the executable file(s) and any support files and/or directories required to run the employer Model Connector. The data directory contains some test files used to construct the Model Connector. The readme.txt file indicates how to execute the application, which is discussed below.

Option 2 is the full State Model Connector project. This download contains a directory that has the executable files and the source files. The State Model Connector project contains all files required to be loaded into the Eclipse IDE or Visual Studio 2010 with minimal adjustments required.

To learn more about the setup and running of the particular technology, please see the section below that corresponds to the technology desired.

7.2.1.1.1 State Requirements for ASCII file

All of the state requirements detailed in Part B of this document are still applicable for the ASCII file. In particular, when creating the ASCII file, you must make sure that it falls under the 8 megabyte limit or it will be rejected by the **Central Broker**. Attachments are handled the same way within the ASCII file as they are within the XML; the software expects attachments to be encoded into the ASCII file. The one minor difference between the XML and ASCII file is that when placing an encoded file into the ASCII file it must be a continuous string with no newline characters in it (must not be chunked). If the ASCII file contains newlines characters in the encoded attachment, the data file reader will not work correctly.

7.2.1.1.2 Request Input

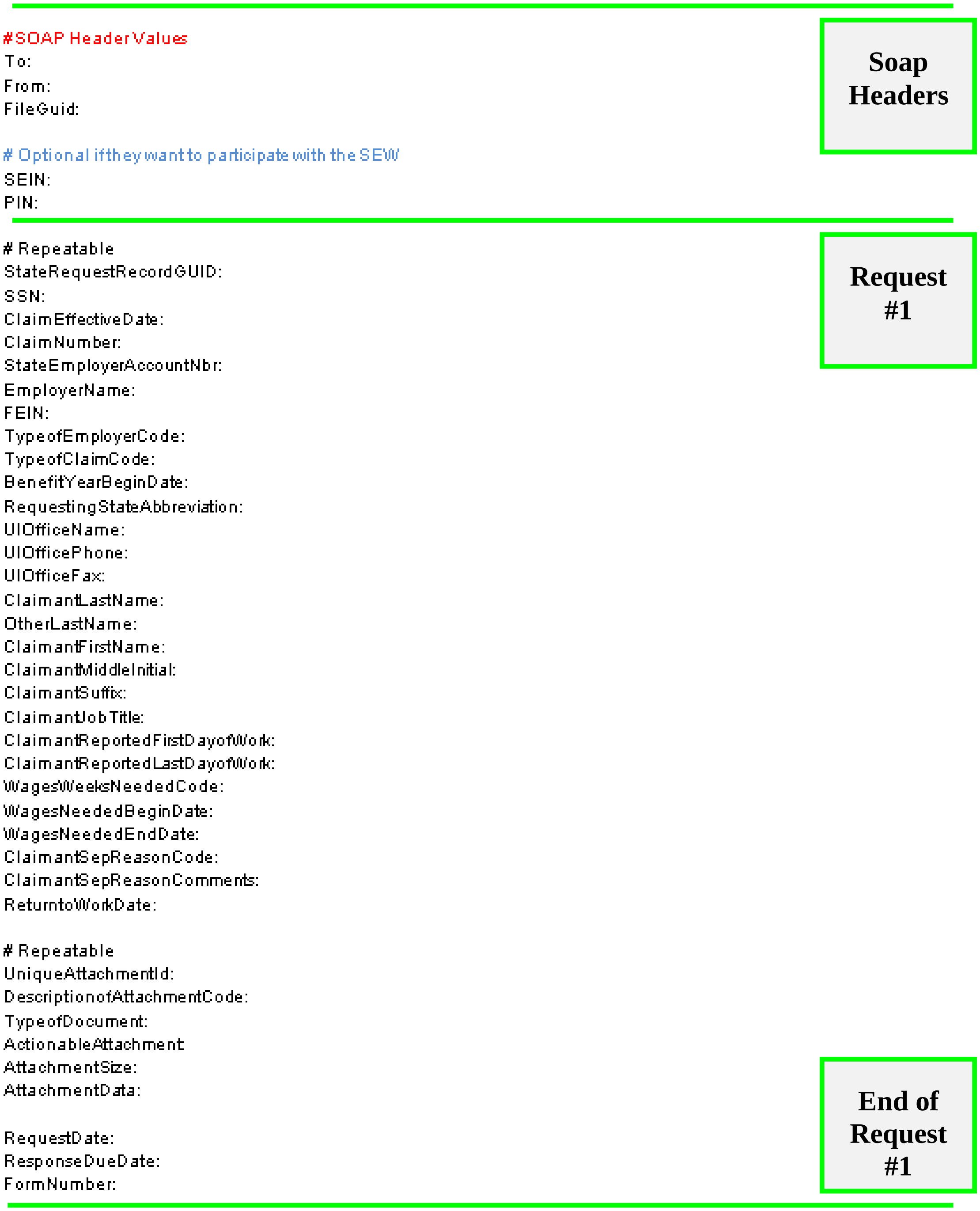
Input files are specified on the command line used to execute the Model Connector.   See examples below.

7.2.1.1.3 Response Output

The SIDES Model Connector can provide responses in ASCII format, PDF, or both ASCII and PDF. In all cases, the responses are available in XML format. ASCII and PDF output files are specified in the runtime configuration parameters. The XML response output file is contained in the results Log file, whose path is specified in the runtime configuration parameters.

7.2.1.1.4 ASCII File Specification – Separation Information Post

The ASCII file ingested by the Model Connector on the Post has two main sections. Section 1 (SOAP Headers) describes the SOAP Headers that must be placed on the SOAP message. The second section (Request) describes the actual request records. These are discussed further below. Any line in the ASCII specification that is empty or contains a # character as the first character is ignored. The # allows comments to be placed in the file.



```
#SOAP Header Values
To:
From:
FileGuid:

# Optional if they want to participate with the SEW
SEIN:
PIN:

# Repeatable
StateRequestRecordGUID:
SSN:
ClaimEffectiveDate:
ClaimNumber:
StateEmployerAccountNbr:
EmployerName:
FEIN:
TypeofEmployerCode:
TypeofClaimCode:
BenefitYearBeginDate:
RequestingStateAbbreviation:
UIOfficeName:
UIOfficePhone:
UIOfficeFax:
ClaimantLastName:
OtherLastName:
ClaimantFirstName:
ClaimantMiddleInitial:
ClaimantSuffix:
ClaimantJobTitle:
ClaimantReportedFirstDayofWork:
ClaimantReportedLastDayofWork:
WagesWeeksNeededCode:
WagesNeededBeginDate:
WagesNeededEndDate:
ClaimantSepReasonCode:
ClaimantSepReasonComments:
ReturntoWorkDate:

# Repeatable
UniqueAttachmentId:
DescriptionofAttachmentCode:
TypeofDocument:
ActionableAttachment
AttachmentSize:
AttachmentData:

RequestDate:
ResponseDueDate:
FormNumber:
```

The SOAP headers section of the ASCII file contains the routing information discussed in section 4.3-SOAP Custom Headers. It must contain the following information:

Table 32 - State Post to Broker

Header Element	Required	Definition	Example
To	Y	The Unique ID of the employer or TPA to which the message is intended For a web services request, the 'To' field will always contain 'BR' followed by nine digits. For a SEW request, the 'To' field will contain the FEIN.	BR000000003
From	Y	The Unique ID of the state where the message originated	UT
FileGUID	Y	The state-generated GUID applied to this message that can uniquely identify this file Size is 32 hexadecimal digits	A42A1FBDAC9549AC7D8D3F45E4040319

For SEW requests, the SOAP header must contain the SEIN and the PIN.

Table 33 - State Post to Broker - SIDES Employer Website

Header Element	Required	Definition	Example
SEIN	N	The SEIN of the employer or TPA to which the message is intended. For those states that do not use the SEIN, this must equal the FEIN Size is up to 20 digits	123456789
PIN	N	The PIN to which the state wants to assign this request for this employer or TPA Size is up to 20 characters	435222169876

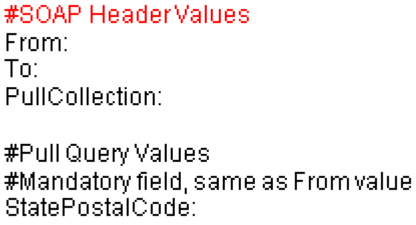
The Request section of the ASCII file contains the actual request record or records. There can be multiple request records in a file. Each request record must start with the element name StateRequestRecordGUID. Within the record itself, there can also be repeatable attachment sections (up to 10 attachments according to the Separation Information specification). The attachment section must contain all of the information on a particular attachment before moving on to the next attachment. Each line in the record contains the Data Element Name as described in the Implementation Guide followed by a colon (:) followed by the value given to that element name. The value must be contained all on a single line. If the data element value is null, it must not be in the ASCII file.

7.2.1.1.5 ASCII File Specification – Separation Information PULL

The ASCII file ingested by the Model Connector on the Pull has two main sections. Section 1 describes the SOAP Headers, which must be placed on the SOAP message. Section 2 describes the Pull Collection Query. These are discussed below. Any line that is empty or has a # character as the first character in it is ignored. The # allows comments to be placed in the file.

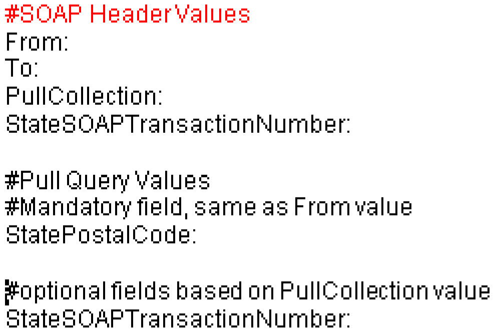
There are three PULL formats that will accomplish the same thing as their XML counterparts described in section 4.3.2-State Pull.

- Regular Pull



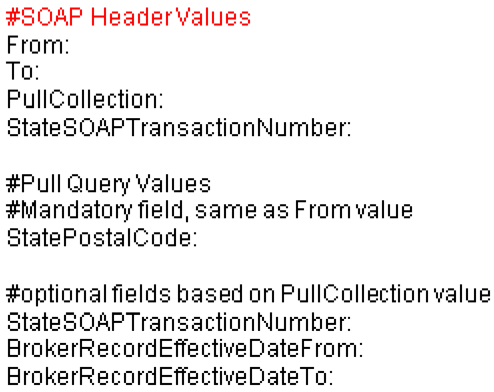
```
#SOAP Header Values
From:
To:
PullCollection:

#Pull Query Values
#Mandatory field, same as From value
StatePostalCode:
```

- Pull By State Soap Transaction Number



```
#SOAP Header Values
From:
To:
PullCollection:
StateSOAPTransactionNumber:

#Pull Query Values
#Mandatory field, same as From value
StatePostalCode:

#optional fields based on PullCollection value
StateSOAPTransactionNumber:
```

- Pull By Date



```
#SOAP Header Values
From:
To:
PullCollection:
StateSOAPTransactionNumber:

#Pull Query Values
#Mandatory field, same as From value
StatePostalCode:

#optional fields based on PullCollection value
StateSOAPTransactionNumber:
BrokerRecordEffectiveDateFrom:
BrokerRecordEffectiveDateTo:
```

If specified in the configuration file, the State Model Connector will return the responses in an ASCII format. The top of the file will contain the SOAP Headers in the response file.

Table 34 - Broker Response to Request (Regular Pull)

Header Element	Definition	Example
To	The Unique ID of the state that requested the Pull	UT
From	The Unique ID of the employer or TPA from which these response records originated	BR000000001
StateSOAPTransactionNumber	The unique number assigned to this file by the Broker	3565
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

After the SOAP headers, the ASCII file will contain all the records returned in the pull call. The format will be the same as described in the post, where there is an element name followed by a colon (:) followed by the value being returned. A complete file specification containing all data elements is not provided as the returned file is dynamic, based upon the business rules. SIDES participants should rely on the standard format to ensure all response values are ingested by the back-end system. The following is an example Response file.

```
#SOAP Headers

To:ST
From:BR999999999
StateSOAPTransactionNumber:142998
MessageCode:1


#Separation Response
StateRequestRecordGUID:30000000000000000000000000004005
BrokerRecordTransactionNumber:2013891
SSN:560348479
ClaimEffectiveDate:2007-06-04
ClaimNumber:378621
StateEmployerAccountNbr:0065560
CorrectedEmployerName:J C Penny
CorrectedStateEmployerAccountNbr:0123456789
CorrectedFEIN:987654321
OtherSSN:660348479
ClaimantNameWorkedAsForEmployer:Charlie Wilson
ClaimantJobTitle:Customer Service Associate
SeasonalEmploymentInd:Y
TotalEarnedWagesNeededInd:2
TotalWeeksWorkedNeededInd:2
AverageWeeklyWage:125.00
EmployerSepReasonCode:5
ReturnToWorkInd:N
ReturnToWorkDate:2010-01-01
WorkingAllAvailableHoursInd:Y

#Remuneration occurence
RemunerationTypeCode:5
RemunerationAmountPerPeriod:999.99
RemunerationPeriodFrequencyCode:W
DateRemunerationIssued:2007-10-15
EmployerAllocationInd:Y
AllocationBeginDate:2007-10-15
AllocationEndDate:2007-10-22

AverageNumberofHoursWorkedperWeek:40
MandatoryPension:N
ContributoryorNotContributoryClaimantInd:Y
ClaimantPensionContributionPercent:100
EmployerSepReasonComments:EmployerSepReasonComments
DischargeReasonCode:5
FinalIncidentReason:FinalIncidentReason
FinalIncidentDate:2007-10-22
ViolateCompanyPolicyInd:N
DischargePolicyAwareExplanationCode:W

WhoDischargedName:Charlie Wilson
```

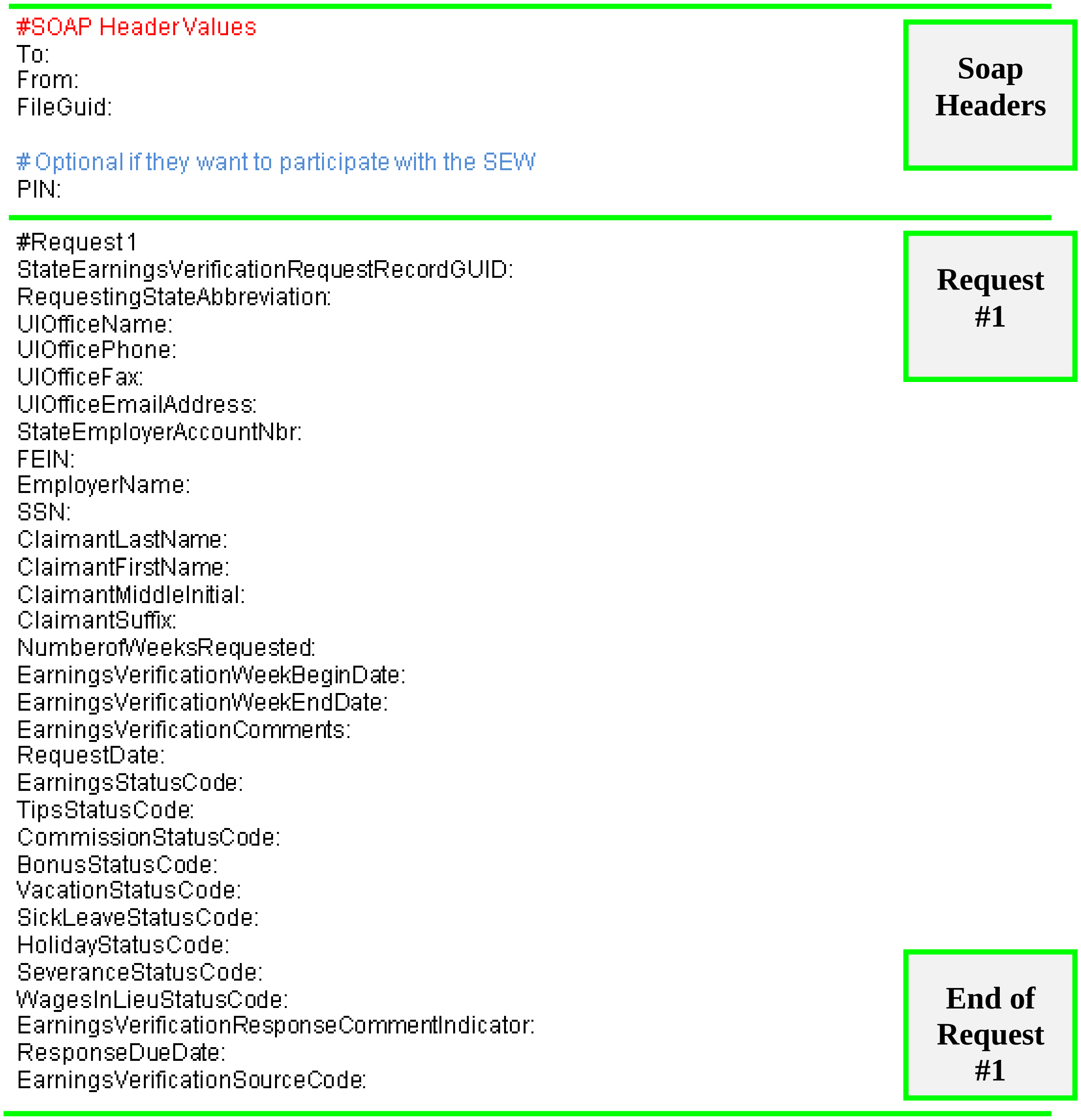
```
WhoDischargedTitle:Customer Service Associate
VoluntarySepReasonCode:5
HiringAgreementChangesCode:5
HiringAgreementChangesComments:HiringAgreementChangesComments
ClaimantActionsToAvoidQuitInd:Y
ActionTakenComments:ActionTakenComments
ContinuingWorkAvailableInd:Y
VoluntarySepReasonComments:The claimant quit without giving JCPenney a
reason.

PreparerTypeCode:E
PreparerCompanyName:J C Penny
PreparerTelephoneNumberPlusExt:9724312108
PreparerContactName:Ed A Jones
PreparerTitle:Project Manager
PreparerFaxNbr:9725312108
PreparerEmailAddress:edjones@jcpenneytest.com
BrokerRecordEffectiveDate:2011-04-08T15:28:41-0400

#Separation Response
StateRequestRecordGUID:30000000000000000000000000004003
BrokerRecordTransactionNumber:2013889
SSN:560348477
ClaimEffectiveDate:2007-06-04
ClaimNumber:388620
StateEmployerAccountNbr:0065560
CorrectedEmployerName:J C Penny
CorrectedStateEmployerAccountNbr:0123456789
CorrectedFEIN:987654321
OtherSSN:660348477
ClaimantNameWorkedAsForEmployer:Andy Wilson
ClaimantJobTitle:Customer Service Associate
SeasonalEmploymentInd:N
EmployerReportedClaimantFirstDayofWork:2007-10-11
EmployerReportedClaimantLastDayofWork:2007-10-14
EffectiveSeparationDate:2007-10-14
TotalEarnedWagesNeededInd:3
TotalWeeksWorkedNeededInd:3
AverageWeeklyWage:125.00
EmployerSepReasonCode:3
ReturnToWorkInd:N
WorkingAllAvailableHoursInd:N
NotWorkingAvailableHoursReason:NotWorkingAvailableHoursReason
LaborDisputeTypeInd:L

#Remuneration occurence
RemunerationTypeCode:3
RemunerationAmountPerPeriod:999.99
RemunerationPeriodFrequencyCode:B
DateRemunerationIssued:2007-10-15
EmployerAllocationInd:N
AllocationBeginDate:2007-10-15
AllocationEndDate:2007-10-22

AverageNumberofHoursWorkedperWeek:40
MandatoryRetirementInd:N
```

```
MandatoryPension:N
ContributoryorNotContributoryClaimantInd:N
ClaimantPensionContributionPercent:100
DischargeReasonCode:3
FinalIncidentReason:FinalIncidentReason
FinalIncidentDate:2007-10-13
ViolateCompanyPolicyInd:N
DischargePolicyAwareInd:N
DischargePolicyAwareExplanationCode:V

#Prior Incident occurence
PriorIncidentDate:2007-10-10
PriorIncidentReason:None
PriorIncidentWarningInd:Y
PriorIncidentWarningDate:2007-10-10
PriorIncidentWarningDescription:Verbal

WhoDischargedName:Andy Wilson
WhoDischargedTitle:Customer Service Associate
VoluntarySepReasonCode:3
HiringAgreementChangesCode:3
HiringAgreementChangesComments:HiringAgreementChangesComments
ClaimantActionsToAvoidQuitInd:N
ContinuingWorkAvailableInd:N

PreparerTypeCode:T
PreparerCompanyName:J C Penny
PreparerTelephoneNumberPlusExt:9724312108
PreparerContactName:Ed A Jones
PreparerTitle:Project Manager
PreparerFaxNbr:9725312108
PreparerEmailAddress:edjones@jcpenneytest.com
BrokerRecordEffectiveDate:2011-04-08T15:28:39-0400

#Separation Response
StateRequestRecordGUID:30000000000000000000000000004004
BrokerRecordTransactionNumber:2013890
SSN:560348478
ClaimEffectiveDate:2007-06-04
ClaimNumber:388620
StateEmployerAccountNbr:0065560
EmployerReportedClaimantFirstDayofWork:2007-10-11
EmployerReportedClaimantLastDayofWork:2007-10-14
EffectiveSeparationDate:2007-10-14
TotalEarnedWagesNeededInd:3
TotalWeeksWorkedNeededInd:3
AverageWeeklyWage:125.00
EmployerSepReasonCode:4
ReturnToWorkInd:Y
ReturnToWorkDate:2010-01-01
WorkingAllAvailableHoursInd:Y

#Remuneration occurence
RemunerationTypeCode:4
RemunerationAmountPerPeriod:999.99
RemunerationPeriodFrequencyCode:M
```

```
DateRemunerationIssued:2007-10-15
EmployerAllocationInd:Y
AllocationBeginDate:2007-10-15
AllocationEndDate:2007-10-22

AverageNumberofHoursWorkedperWeek:40
MandatoryRetirementInd:N
MandatoryPension:N
ContributoryorNotContributoryClaimantInd:Y
ClaimantPensionContributionPercent:100
EmployerSepReasonComments:EmployerSepReasonComments
DischargeReasonCode:4
FinalIncidentReason:FinalIncidentReason
FinalIncidentDate:2007-10-13
ViolateCompanyPolicyInd:Y
DischargePolicyAwareInd:Y
DischargePolicyAwareExplanationCode:W

#Prior Incident occurence
PriorIncidentDate:2007-10-10
PriorIncidentReason:None
PriorIncidentWarningInd:Y
PriorIncidentWarningDate:2007-10-10
PriorIncidentWarningDescription:Verbal

WhoDischargedName:Brian Wilson
WhoDischargedTitle:Customer Service Associate
VoluntarySepReasonCode:4
HiringAgreementChangesCode:4
HiringAgreementChangesComments:HiringAgreementChangesComments
ClaimantActionsToAvoidQuitInd:Y
ActionTakenComments:ActionTakenComments
ContinuingWorkAvailableInd:Y

PreparerTypeCode:E
PreparerCompanyName:J C Penny
PreparerTelephoneNumberPlusExt:9724312108
PreparerContactName:Ed A Jones
PreparerTitle:Project Manager
PreparerFaxNbr:9725312108
PreparerEmailAddress:edjones@jcpenneytest.com
BrokerRecordEffectiveDate:2011-04-08T15:28:40-0400
```

7.2.1.1.6 ASCII File Specification – Earnings Verification POST

The ASCII file ingested by the Model Connector on the Post has two main sections to it. Section 1 describes the SOAP Headers that must be placed on the SOAP message. Section 2 describes the actual request record(s). These are discussed below. Any line that is empty or has a # character as the first character in it is ignored. The # allows comments to be placed in the file.



```
#SOAP Header Values
To:
From:
FileGuid:

# Optional if they want to participate with the SEW
PIN:
```

```
#Request 1
StateEarningsVerificationRequestRecordGUID:
RequestingStateAbbreviation:
UIOfficeName:
UIOfficePhone:
UIOfficeFax:
UIOfficeEmailAddress:
StateEmployerAccountNbr:
FEIN:
EmployerName:
SSN:
ClaimantLastName:
ClaimantFirstName:
ClaimantMiddleInitial:
ClaimantSuffix:
NumberofWeeksRequested:
EarningsVerificationWeekBeginDate:
EarningsVerificationWeekEndDate:
EarningsVerificationComments:
RequestDate:
EarningsStatusCode:
TipsStatusCode:
CommissionStatusCode:
BonusStatusCode:
VacationStatusCode:
SickLeaveStatusCode:
HolidayStatusCode:
SeveranceStatusCode:
WagesInLieuStatusCode:
EarningsVerificationResponseCommentIndicator:
ResponseDueDate:
EarningsVerificationSourceCode:
```

The first part of the ASCII file contains the SOAP headers. This is the routing information discussed in section 4.3-SOAP Custom Headers. It must contain the following information:

Table 35 - State Post to Broker

Header Element	Required	Definition	Example
To	Y	The Unique ID of the employer or TPA to which the message is intended For a web services request, the 'To' field will always contain 'BR' followed by	BR000000003

Header Element	Required	Definition	Example
		nine digits. For a SEW request, the 'To' field will contain the FEIN.	
From	Y	The Unique ID of the state where the message originated	UT
FileGUID	Y	The state-generated GUID applied to this message that can uniquely identify this file Size is 32 hexadecimal digits	A42A1FBDAC9549AC7D8D3F45E4040319

For SEW requests, the PIN is required.

Table 36 - State Post to Broker - SIDES Employer Website

Header Element	Required	Definition	Example
PIN	N	The PIN to which the state wants to assign this request for this employer or TPA Size is up to 20 characters	435222169876

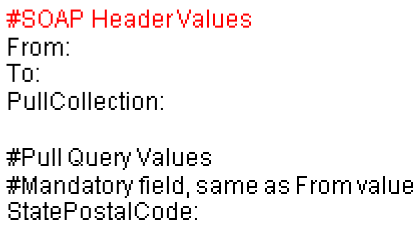
The second part of the ASCII file is the actual request record(s). There can be multiple request records in a file. Each request record must begin with the element name StateEarningsVerificationRequestRecordGUID. Each line in the record contains the Data Element Name as described in the Implementation Guide followed by a colon (:) followed by the value given to that element name. The value must be contained all on a single line. If the data element value is null, it must not be in the ASCII file.

7.2.1.1.7 ASCII File Specification – Earnings Verification PULL

The ASCII file ingested by the Model Connector on the Pull has two main sections to it. The first section describes the SOAP Headers that must be placed on the SOAP message. The second section describes the Pull Collection Query. These are discussed below. Any line that is empty or has a # character as the first character in it is ignored. The # allows comments to be placed in the file.

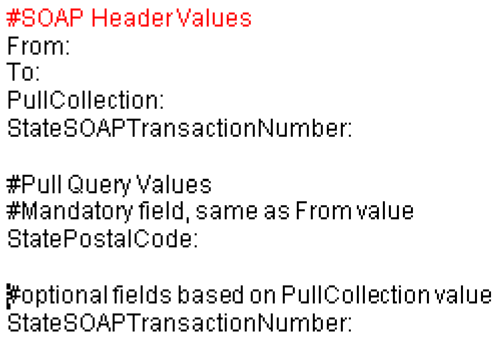
There are three PULL formats that will accomplish the same thing as their XML counterparts described in section 4.3.2-State Pull.

- Regular Pull



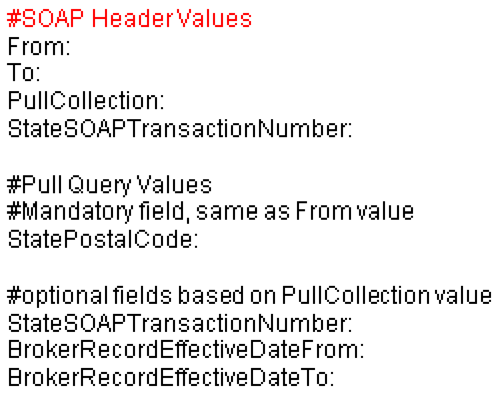
```
#SOAP Header Values
From:
To:
PullCollection:

#Pull Query Values
#Mandatory field, same as From value
StatePostalCode:
```

- Pull By State Soap Transaction Number



```
#SOAP Header Values
From:
To:
PullCollection:
StateSOAPTransactionNumber:

#Pull Query Values
#Mandatory field, same as From value
StatePostalCode:

#optional fields based on PullCollection value
StateSOAPTransactionNumber:
```

- Pull By Date



```
#SOAP Header Values
From:
To:
PullCollection:
StateSOAPTransactionNumber:

#Pull Query Values
#Mandatory field, same as From value
StatePostalCode:

#optional fields based on PullCollection value
StateSOAPTransactionNumber:
BrokerRecordEffectiveDateFrom:
BrokerRecordEffectiveDateTo:
```

If specified in the configuration file, the State Model Connector will return the responses in an ASCII format. The top of the file will contain the SOAP Headers in the response file.

Table 37 - Broker Response to Request (Regular Pull)

Header Element	Definition	Example
To	The Unique ID of the state that requested the Pull	UT
From	The Unique ID of the employer or TPA from which these response records originated	BR000000001

Header Element	Definition	Example
StateSOAPTransactionNumber	The unique number assigned to this file by the Broker	3565
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

After the SOAP headers, the ASCII file will contain all the records returned in the pull call. The format will be the same as described in the post, where there is an element name followed by a colon (:) followed by the value being returned. A complete file specification containing all data elements is not provided as the returned file is dynamic, based upon the business rules and request indicator values. SIDES participants should rely on the standard format to ensure all response values are ingested by the back-end system. The following is an example Response file.

```
#SOAP Headers
To:ST
From:BR999999999
StateSOAPTransactionNumber:9217
MessageCode:1

#Earnings Verification Response
StateEarningsVerificationRequestRecordGUID:AAA51000000000000000000000000001
BrokerRecordTransactionNumber:5445
RequestingStateAbbreviation:ST
UIOfficeName:Office Name
StateEmployerAccountNbr:1234567890
FEIN:123456789
CorrectedFEIN:987654321
EmployerName:ACME
CorrectedEmployerName:Fly By Night
SSN:211111111
ClaimantNameWorkedAsForEmployer:John Q Public
NumberofWeeksRequested:5
EarningsVerificationWeekBeginDate:2010-08-01
EarningsVerificationWeekEndDate:2010-09-04
ClaimantEmployerWorkRelationshipCode:John Q Public
EmployerEarningsCode:1
FirstDayWorkedinPeriod:2010-08-01
StillWorkingCode:2
LastDayWorked:2010-09-04
EmployerSepReasonCode:1
```

```
EarningsVerificationResponseComment:This employee was let go during the time
period

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-01
WeekEndDate:2010-08-07
HoursWorked:15:00
AmountEarnedForWeek:500.00

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-08
WeekEndDate:2010-08-14
HoursWorked:15:00
AmountEarnedForWeek:500.00

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-15
WeekEndDate:2010-08-21
HoursWorked:15:00
AmountEarnedForWeek:500.00

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-22
WeekEndDate:2010-08-28
HoursWorked:101:00
AmountEarnedForWeek:500.00

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-29
WeekEndDate:2010-09-04
HoursWorked:5:00
AmountEarnedForWeek:500.00

PreparerTypeCode:T
PreparerCompanyName:ABC TPA
PreparerTelephoneNumberPlusExt:5555555556
PreparerContactName:Mrs Sue Herman
PreparerTitle:Claims Administrator
PreparerFaxNbr:5555555557
PreparerEmailAddress:sue.herman@abctpa.com
EarningsVerificationSourceCode:9
BrokerRecordEffectiveDate:2011-04-08T15:51:50-0400

#Earnings Verification Response
StateEarningsVerificationRequestRecordGUID:AAA52000000000000000000000000002
BrokerRecordTransactionNumber:5446
RequestingStateAbbreviation:ST
UIOfficeName:Office Name
StateEmployerAccountNbr:1234567890
FEIN:123456789
CorrectedFEIN:987654321
EmployerName:ACME
CorrectedEmployerName:Fly By Night
SSN:211121314
ClaimantNameWorkedAsForEmployer:John Q Public
NumberofWeeksRequested:5
```

```
EarningsVerificationWeekBeginDate:2010-08-01
EarningsVerificationWeekEndDate:2010-09-04
ClaimantEmployerWorkRelationshipCode:John Q Public
EmployerEarningsCode:1
FirstDayWorkedinPeriod:2010-08-01
StillWorkingCode:2
LastDayWorked:2010-09-04
EmployerSepReasonCode:1
EarningsVerificationResponseComment:This employee was let go during the time
period

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-01
WeekEndDate:2010-08-07
HoursWorked:15:00
AmountEarnedForWeek:500.00
VacationAmountPaidForWeek:40.00
HolidayAmountPaidForWeek:60.00
HolidayPaidDate:2010-08-07
WagesInLieuAmountPaidForWeek:80.00
WagesInLieuPaidDate:2010-08-07

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-08
WeekEndDate:2010-08-14
HoursWorked:15:00
AmountEarnedForWeek:500.00
VacationAmountPaidForWeek:40.00
HolidayAmountPaidForWeek:60.00
HolidayPaidDate:2010-08-14
WagesInLieuAmountPaidForWeek:80.00
WagesInLieuPaidDate:2010-08-14

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-15
WeekEndDate:2010-08-21
HoursWorked:15:00
AmountEarnedForWeek:500.00
VacationAmountPaidForWeek:40.00
HolidayAmountPaidForWeek:60.00
HolidayPaidDate:2010-08-21
WagesInLieuAmountPaidForWeek:80.00
WagesInLieuPaidDate:2010-08-21

#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-22
WeekEndDate:2010-08-28
HoursWorked:101:00
AmountEarnedForWeek:500.00
VacationAmountPaidForWeek:40.00
HolidayAmountPaidForWeek:60.00
HolidayPaidDate:2010-08-28
WagesInLieuAmountPaidForWeek:80.00
WagesInLieuPaidDate:2010-08-28
```

```
#Weekly Earnings Verification occurence
WeekBeginDate:2010-08-29
WeekEndDate:2010-09-04
HoursWorked:5:00
AmountEarnedForWeek:500.00
VacationAmountPaidForWeek:40.00
HolidayAmountPaidForWeek:60.00
HolidayPaidDate:2010-09-04
WagesInLieuAmountPaidForWeek:80.00
WagesInLieuPaidDate:2010-09-04

PreparerTypeCode:T
PreparerCompanyName:ABC TPA
PreparerTelephoneNumberPlusExt:5555555556
PreparerContactName:Mrs Sue Herman
PreparerTitle:Claims Administrator
PreparerFaxNbr:5555555557
PreparerEmailAddress:sue.herman@abctpa.com
EarningsVerificationSourceCode:9
BrokerRecordEffectiveDate:2011-04-08T15:51:52-0400
```

7.2.1.2 Log Files – POST

7.2.1.2.1 DEBUG log file

This log file is the main debugging log file for the whole application for a given run. It contains all debug output logged in the system during that run. If the system were to fail unexpectedly, this log file will contain the most up to date status and will most likely indicate where the system failed. It also includes all the data that is written to the other log files.

The log file is placed into the directory specified by the DebugLogFilePath configuration parameter. The naming scheme is as follows:

```
POST_{SI|EV}_DEBUG_date_time.log

For example:

POST_EV_DEBUG_2011-04-08_15-23-18-292.log
```

7.2.1.2.2 BRPT log file

This log file shows the results from the call to the BRPT on the request files submitted for a given run. It will indicate all the records that had a problem in them and were thus stripped for the request file being sent. It is the responsibility of the State to correct these errors and retransmit these requests to the **Central Broker**.

Here is an example of the contents of a BRPT log file:

```
#Failed Records
Record GUID Failure:30000000000000000000000000004001
```

```
Number of errors detected:2
#Errors
Error Number:1
Error Code:111
Error Message:Business Rule violation - There must be a value (Date) for
WagesNeededBeginDate if WagesWeeksNeededCode = WO|WW
Error Number:2
Error Code:102
Error Message:Business Rule violation - Two or more UniqueAttachmentIDs
assigned to a specific Separation Information Request are the same - they
must be unique.
```

The log file is placed into the directory specified by the BrptLogFilePath configuration parameter. The naming scheme is as follows:

```
POST_{SI|EV}_BRPT_{to}_{from}_date_time_guid.log

For example:

POST_EV_BRPT_BR999999999_ST_2011-04-08_15-23-18-
292_01234567890123456789012345678901.log
```

7.2.1.2.3 RESULTS log file

This log file shows the acknowledgement from the **Central Broker** for a given run.

Here is an example of the contents of a RESULTS log file.

```
#SOAP Headers in Acknowledgement
To:ST
StateRequestFileGUID:01234567890123456789012345678901
From:Broker
MessageCode:1
#Acknowledgement
File GUID:01234567890123456789012345678901
Number of Request Records Received:1
Number of Request Records Received in Error:0
Date Started Received:2011-04-09T10:07:03.257-04:00
Date Finished Receiving:2011-04-09T10:07:03.601-04:00
```

The log file is placed into the directory specified by the ResultsLogFilePath configuration parameter. The naming scheme is as follows:

```
POST_{SI|EV}_RESULTS_{to}_{from}_date_time_guid.log

For example:

POST_SI_RESULTS_BR999999999_ST_2011-04-08_15-23-18-
292_01234567890123456789012345678901.log
```

7.2.1.2.4 PIN log file

This log file shows the created PIN that was used in the call to the SEW. The PIN log file will only exist if a pin was created. If the config file had the createPin config parameter set to false or there was a problem with the SOAP headers (i.e. the To: header specified was not a 9 digit FEIN) the PIN log file will not be created. If you expect to see a PIN log file but one was not created, view the debug file for that run which will give more detailed information.

Here is an example of the contents of a PIN log file.

```
PIN:20110409120056488
```

The log file is placed into the directory specified by the PINLogFilePath configuration parameter. The naming scheme is as follows:

```
POST_{SI|EV}_PIN_{to}_{from}_date_time_guid.log

For example:

POST_SI_PIN_BR999999999_ST_2011-04-08_15-23-18-
292_01234567890123456789012345678901.log
```

7.2.1.3 Log Files – PULL

7.2.1.3.1 DEBUG log file

This is the main debugging log file for the whole application for a given run. It contains all debug output logged in the system during that run. If the system were to fail unexpectedly, this log file will contain the most up to date status and will most likely indicate where the system failed. It also includes all the data written to the other log files.

The log file is placed into the directory specified by the DebugLogFilePath configuration parameter. The naming scheme is as follows:

```
PULL_{SI|EV}_DEBUG_date_time.log

For example:

PULL_EV_DEBUG_2011-04-08_15-46-12-035.log
```

7.2.1.3.2 BRPT log file

This log file shows the results from the call to the BRPT on the response files returned by the **Central Broker** for a given run. If there are any records in this file, then the State Model

Connector will return a Message Code of 2 back to the **Central Broker** indicating a failure. This file will then be pulled again on its next Pull call.

Here is an example of the contents of a BRPT log file:

```
#Failed Records
Record GUID Failure:3000000000000000000000000004000
Number of errors detected: 1
#Errors
Error Number:1
Error Code:201
Error Message:XSD validation violation
```

The log file is placed into the directory specified by the BrptLogFilePath configuration parameter. The naming scheme is as follows:

```
PULL_{SI|EV}_BRPT_{to}_{from}_date_time_PullCollection.log

For example:

PULL_EV_BRPT_Broker_ST_2011-04-08_15-46-12-035_1.log
```

7.2.1.3.3 RESULTS log file

This log file shows the results of the Pull call from the **Central Broker** for a given run. It will contain the State Separation Responses in XML format. If the pullAllFiles config file parameter is set to true, then this file will contain all of the SOAP Headers and Response Payloads the Model Connector received from the **Central Broker**.

Here is an example of the contents of a RESULTS log file.

```
#SOAP Headers
To:ST
From:BR999999999
StateSOAPTransactionNumber:143104
MessageCode:1

#Response Payload
<StateSeparationResponseCollection xmlns:ns2="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-ws        -utility-1.0.xsd"
xmlns="https:// REDACTED /schemas">
    <StateSeparationResponse>

      <StateRequestRecordGUID>3000000000000000000000000004004</StateRequestR
      ecordGUID>
      <BrokerRecordTransactionNumber>2013890</BrokerRecordTransactionNumber>
        <SSN>560348478</SSN>
        <ClaimEffectiveDate>2007-06-04</ClaimEffectiveDate>
        <ClaimNumber>388620</ClaimNumber>
        <StateEmployerAccountNbr>0065560</StateEmployerAccountNbr>
        <EmployerReportedClaimantFirstDayofWork>2007-10-
11</EmployerReportedClaimantFirstDayofWork>
        <EmployerReportedClaimantLastDayofWork>2007-10-
14</EmployerReportedClaimantLastDayofWork>
        <EffectiveSeparationDate>2007-10-14</EffectiveSeparationDate>
        <TotalEarnedWagesNeededInd>3</TotalEarnedWagesNeededInd>
        <TotalWeeksWorkedNeededInd>3</TotalWeeksWorkedNeededInd>
        <AverageWeeklyWage>125.00</AverageWeeklyWage>
        <EmployerSepReasonCode>4</EmployerSepReasonCode>
        <ReturnToWorkInd>Y</ReturnToWorkInd>
        <ReturnToWorkDate>2010-01-01</ReturnToWorkDate>
        <WorkingAllAvailableHoursInd>Y</WorkingAllAvailableHoursInd>
        <Remuneration>
            <RemunerationTypeCode>4</RemunerationTypeCode>
            <RemunerationAmountPerPeriod>999.99</RemunerationAmountPerPeriod>

<RemunerationPeriodFrequencyCode>M</RemunerationPeriodFrequencyCode>
            <DateRemunerationIssued>2007-10-15</DateRemunerationIssued>
            <EmployerAllocationInd>Y</EmployerAllocationInd>
            <AllocationBeginDate>2007-10-15</AllocationBeginDate>
            <AllocationEndDate>2007-10-22</AllocationEndDate>
        </Remuneration>

<AverageNumberofHoursWorkedperWeek>40</AverageNumberofHoursWorkedperWeek>
        <MandatoryRetirementInd>N</MandatoryRetirementInd>
        <MandatoryPension>N</MandatoryPension>

<ContributoryorNotContributoryClaimantInd>Y</ContributoryorNotContributoryCla
imantInd>

<ClaimantPensionContributionPercent>100</ClaimantPensionContributionPercent>

<EmployerSepReasonComments>EmployerSepReasonComments</EmployerSepReasonCommen
ts>
        <DischargeReasonCode>4</DischargeReasonCode>
        <FinalIncidentReason>FinalIncidentReason</FinalIncidentReason>
        <FinalIncidentDate>2007-10-13</FinalIncidentDate>
        <ViolateCompanyPolicyInd>Y</ViolateCompanyPolicyInd>
        <DischargePolicyAwareInd>Y</DischargePolicyAwareInd>

<DischargePolicyAwareExplanationCode>W</DischargePolicyAwareExplanationCode>
        <PriorIncidentOccurrence>
            <PriorIncidentDate>2007-10-10</PriorIncidentDate>
            <PriorIncidentReason>None</PriorIncidentReason>
            <PriorIncidentWarningInd>Y</PriorIncidentWarningInd>
            <PriorIncidentWarningDate>2007-10-10</PriorIncidentWarningDate>

<PriorIncidentWarningDescription>Verbal</PriorIncidentWarningDescription>
        </PriorIncidentOccurrence>
        <WhoDischargedName>Brian Wilson</WhoDischargedName>
        <WhoDischargedTitle>Customer Service Associate</WhoDischargedTitle>
        <VoluntarySepReasonCode>4</VoluntarySepReasonCode>
        <HiringAgreementChangesCode>4</HiringAgreementChangesCode>
```

```
<HiringAgreementChangesComments>HiringAgreementChangesComments</HiringAgreeme
ntChangesComments>
        <ClaimantActionsToAvoidQuitInd>Y</ClaimantActionsToAvoidQuitInd>
        <ActionTakenComments>ActionTakenComments</ActionTakenComments>
        <ContinuingWorkAvailableInd>Y</ContinuingWorkAvailableInd>
        <PreparerTypeCode>E</PreparerTypeCode>
        <PreparerCompanyName>J C Penny</PreparerCompanyName>

<PreparerTelephoneNumberPlusExt>9724312108</PreparerTelephoneNumberPlusExt>
        <PreparerContactName>Ed A Jones</PreparerContactName>
        <PreparerTitle>Project Manager</PreparerTitle>
        <PreparerFaxNbr>9725312108</PreparerFaxNbr>
        <PreparerEmailAddress>edjones@jcpenneytest.com</PreparerEmailAddress>
        <BrokerRecordEffectiveDate>2011-04-09T12:24:47.000-
04:00</BrokerRecordEffectiveDate>
    </StateSeparationResponse>
    <StateSeparationResponse>

<StateRequestRecordGUID>30000000000000000000000000004003</StateRequestRecordG
UID>

<BrokerRecordTransactionNumber>2013889</BrokerRecordTransactionNumber>
        <SSN>560348477</SSN>
        <ClaimEffectiveDate>2007-06-04</ClaimEffectiveDate>
        <ClaimNumber>388620</ClaimNumber>
        <StateEmployerAccountNbr>0065560</StateEmployerAccountNbr>
        <CorrectedEmployerName>J C Penny</CorrectedEmployerName>

<CorrectedStateEmployerAccountNbr>0123456789</CorrectedStateEmployerAccountNb
r>
        <CorrectedFEIN>987654321</CorrectedFEIN>
        <OtherSSN>660348477</OtherSSN>
        <ClaimantNameWorkedAsForEmployer>Andy
Wilson</ClaimantNameWorkedAsForEmployer>
        <ClaimantJobTitle>Customer Service Associate</ClaimantJobTitle>
        <SeasonalEmploymentInd>N</SeasonalEmploymentInd>
        <EmployerReportedClaimantFirstDayofWork>2007-10-
11</EmployerReportedClaimantFirstDayofWork>
        <EmployerReportedClaimantLastDayofWork>2007-10-
14</EmployerReportedClaimantLastDayofWork>
        <EffectiveSeparationDate>2007-10-14</EffectiveSeparationDate>
        <TotalEarnedWagesNeededInd>3</TotalEarnedWagesNeededInd>
        <TotalWeeksWorkedNeededInd>3</TotalWeeksWorkedNeededInd>
        <AverageWeeklyWage>125.00</AverageWeeklyWage>
        <EmployerSepReasonCode>3</EmployerSepReasonCode>
        <ReturnToWorkInd>N</ReturnToWorkInd>
        <WorkingAllAvailableHoursInd>N</WorkingAllAvailableHoursInd>

<NotWorkingAvailableHoursReason>NotWorkingAvailableHoursReason</NotWorkingAva
ilableHoursReason>
        <LaborDisputeTypeInd>L</LaborDisputeTypeInd>
        <Remuneration>
            <RemunerationTypeCode>3</RemunerationTypeCode>
            <RemunerationAmountPerPeriod>999.99</RemunerationAmountPerPeriod>
```

```
<RemunerationPeriodFrequencyCode>B</RemunerationPeriodFrequencyCode>
            <DateRemunerationIssued>2007-10-15</DateRemunerationIssued>
            <EmployerAllocationInd>N</EmployerAllocationInd>
            <AllocationBeginDate>2007-10-15</AllocationBeginDate>
            <AllocationEndDate>2007-10-22</AllocationEndDate>
        </Remuneration>

<AverageNumberofHoursWorkedperWeek>40</AverageNumberofHoursWorkedperWeek>
        <MandatoryRetirementInd>N</MandatoryRetirementInd>
        <MandatoryPension>N</MandatoryPension>

<ContributoryorNotContributoryClaimantInd>N</ContributoryorNotContributoryCla
imantInd>

<ClaimantPensionContributionPercent>100</ClaimantPensionContributionPercent>
        <DischargeReasonCode>3</DischargeReasonCode>
        <FinalIncidentReason>FinalIncidentReason</FinalIncidentReason>
        <FinalIncidentDate>2007-10-13</FinalIncidentDate>
        <ViolateCompanyPolicyInd>N</ViolateCompanyPolicyInd>
        <DischargePolicyAwareInd>N</DischargePolicyAwareInd>

<DischargePolicyAwareExplanationCode>V</DischargePolicyAwareExplanationCode>
        <PriorIncidentOccurrence>
            <PriorIncidentDate>2007-10-10</PriorIncidentDate>
            <PriorIncidentReason>None</PriorIncidentReason>
            <PriorIncidentWarningInd>Y</PriorIncidentWarningInd>
            <PriorIncidentWarningDate>2007-10-10</PriorIncidentWarningDate>

<PriorIncidentWarningDescription>Verbal</PriorIncidentWarningDescription>
        </PriorIncidentOccurrence>
        <WhoDischargedName>Andy Wilson</WhoDischargedName>
        <WhoDischargedTitle>Customer Service Associate</WhoDischargedTitle>
        <VoluntarySepReasonCode>3</VoluntarySepReasonCode>
        <HiringAgreementChangesCode>3</HiringAgreementChangesCode>

<HiringAgreementChangesComments>HiringAgreementChangesComments</HiringAgreeme
ntChangesComments>
        <ClaimantActionsToAvoidQuitInd>N</ClaimantActionsToAvoidQuitInd>
        <ContinuingWorkAvailableInd>N</ContinuingWorkAvailableInd>
        <PreparerTypeCode>T</PreparerTypeCode>
        <PreparerCompanyName>J C Penny</PreparerCompanyName>

<PreparerTelephoneNumberPlusExt>9724312108</PreparerTelephoneNumberPlusExt>
        <PreparerContactName>Ed A Jones</PreparerContactName>
        <PreparerTitle>Project Manager</PreparerTitle>
        <PreparerFaxNbr>9725312108</PreparerFaxNbr>
        <PreparerEmailAddress>edjones@jcpenneytest.com</PreparerEmailAddress>
        <BrokerRecordEffectiveDate>2011-04-09T12:24:44.000-
04:00</BrokerRecordEffectiveDate>
    </StateSeparationResponse>
    <StateSeparationResponse>

<StateRequestRecordGUID>3000000000000000000000000004005</StateRequestRecordG
UID>
```

```
<BrokerRecordTransactionNumber>2013891</BrokerRecordTransactionNumber>
        <SSN>560348479</SSN>
        <ClaimEffectiveDate>2007-06-04</ClaimEffectiveDate>
        <ClaimNumber>378621</ClaimNumber>
        <StateEmployerAccountNbr>0065560</StateEmployerAccountNbr>
        <CorrectedEmployerName>J C Penny</CorrectedEmployerName>

<CorrectedStateEmployerAccountNbr>0123456789</CorrectedStateEmployerAccountNb
r>
        <CorrectedFEIN>987654321</CorrectedFEIN>
        <OtherSSN>660348479</OtherSSN>
        <ClaimantNameWorkedAsForEmployer>Charlie
Wilson</ClaimantNameWorkedAsForEmployer>
        <ClaimantJobTitle>Customer Service Associate</ClaimantJobTitle>
        <SeasonalEmploymentInd>Y</SeasonalEmploymentInd>
        <TotalEarnedWagesNeededInd>2</TotalEarnedWagesNeededInd>
        <TotalWeeksWorkedNeededInd>2</TotalWeeksWorkedNeededInd>
        <AverageWeeklyWage>125.00</AverageWeeklyWage>
        <EmployerSepReasonCode>5</EmployerSepReasonCode>
        <ReturnToWorkInd>N</ReturnToWorkInd>
        <ReturnToWorkDate>2010-01-01</ReturnToWorkDate>
        <WorkingAllAvailableHoursInd>Y</WorkingAllAvailableHoursInd>
        <Remuneration>
            <RemunerationTypeCode>5</RemunerationTypeCode>
            <RemunerationAmountPerPeriod>999.99</RemunerationAmountPerPeriod>

<RemunerationPeriodFrequencyCode>W</RemunerationPeriodFrequencyCode>
            <DateRemunerationIssued>2007-10-15</DateRemunerationIssued>
            <EmployerAllocationInd>Y</EmployerAllocationInd>
            <AllocationBeginDate>2007-10-15</AllocationBeginDate>
            <AllocationEndDate>2007-10-22</AllocationEndDate>
        </Remuneration>

<AverageNumberofHoursWorkedperWeek>40</AverageNumberofHoursWorkedperWeek>
        <MandatoryPension>N</MandatoryPension>

<ContributoryorNotContributoryClaimantInd>Y</ContributoryorNotContributoryCla
imantInd>

<ClaimantPensionContributionPercent>100</ClaimantPensionContributionPercent>

<EmployerSepReasonComments>EmployerSepReasonComments</EmployerSepReasonCommen
ts>
        <DischargeReasonCode>5</DischargeReasonCode>
        <FinalIncidentReason>FinalIncidentReason</FinalIncidentReason>
        <FinalIncidentDate>2007-10-22</FinalIncidentDate>
        <ViolateCompanyPolicyInd>N</ViolateCompanyPolicyInd>

<DischargePolicyAwareExplanationCode>W</DischargePolicyAwareExplanationCode>
        <WhoDischargedName>Charlie Wilson</WhoDischargedName>
        <WhoDischargedTitle>Customer Service Associate</WhoDischargedTitle>
        <VoluntarySepReasonCode>5</VoluntarySepReasonCode>
        <HiringAgreementChangesCode>5</HiringAgreementChangesCode>
```

```
<HiringAgreementChangesComments>HiringAgreementChangesComments</HiringAgreeme
ntChangesComments>
        <ClaimantActionsToAvoidQuitInd>Y</ClaimantActionsToAvoidQuitInd>
        <ActionTakenComments>ActionTakenComments</ActionTakenComments>
        <ContinuingWorkAvailableInd>Y</ContinuingWorkAvailableInd>
        <VoluntarySepReasonComments>The claimant quit without giving JCPenney
a reason.</VoluntarySepReasonComments>
        <PreparerTypeCode>E</PreparerTypeCode>
        <PreparerCompanyName>J C Penny</PreparerCompanyName>

<PreparerTelephoneNumberPlusExt>9724312108</PreparerTelephoneNumberPlusExt>
        <PreparerContactName>Ed A Jones</PreparerContactName>
        <PreparerTitle>Project Manager</PreparerTitle>
        <PreparerFaxNbr>9725312108</PreparerFaxNbr>
        <PreparerEmailAddress>edjones@jcpenneytest.com</PreparerEmailAddress>
        <BrokerRecordEffectiveDate>2011-04-09T12:24:55.000-
04:00</BrokerRecordEffectiveDate>
    </StateSeparationResponse>
</StateSeparationResponseCollection>

#SOAP Headers
To:ST
From:Broker
StateSOAPTransactionNumber:143105
MessageCode:2

#Response Payload
<StateSeparationResponseCollection xmlns:ns2="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-ws        -utility-1.0.xsd"
xmlns="https:// REDACTED /schemas"/>
```

The log file is placed into the directory specified by the ResultsLogFilePath configuration parameter. The naming scheme is as follows:

```
PULL_{SI|EV}_RESULTS_{to}_{from}_date_time_PullCollection.log

For example:

PULL_SI_RESULTS_Broker_ST_2011-04-08_15-46-12-035_1.log
```

7.2.1.4 Setup Employer Model Connector

To begin using the Employer Model Connector, the employer or TPA must first download the software from the sides.itsc.org website.

There are two options for download of the Employer Model Connector.

The first option is the ‘Black Box’ approach. This download contains only the executable file(s) and any support files and/or directories required to run the employer Model Connector. The data directory contains some test files used to construct the Model Connector. The readme.txt file indicates how to execute the application, which is discussed below.

Option 2 is the full Employer Model Connector project. This download contains a directory that has the executable files and the source files. The Employer Model Connector project contains all files required to be loaded into the Eclipse IDE or Visual Studio 2010 with minimal adjustments required.

To learn more about the setup and running of the particular technology, please see the section below that corresponds to the technology desired.

7.2.1.4.1 Employer Requirements for ASCII file

All of the employer/TPA requirements detailed in Part B of this document are still applicable for the ASCII file. In particular, when creating the ASCII file, you must make sure that it falls under the 8 megabyte limit or it will be rejected by the **Central Broker**. Attachments are handled the same way within the ASCII file as they are within the XML; the software expects attachments to be encoded into the ASCII file. The one minor difference between the XML and ASCII file is that when placing an encoded file into the ASCII file it must be a continuous string with no newline characters in it (must not be chunked). If the ASCII file contains newlines characters in the encoded attachment, the data file reader will not work correctly.

7.2.1.4.2 Response Input

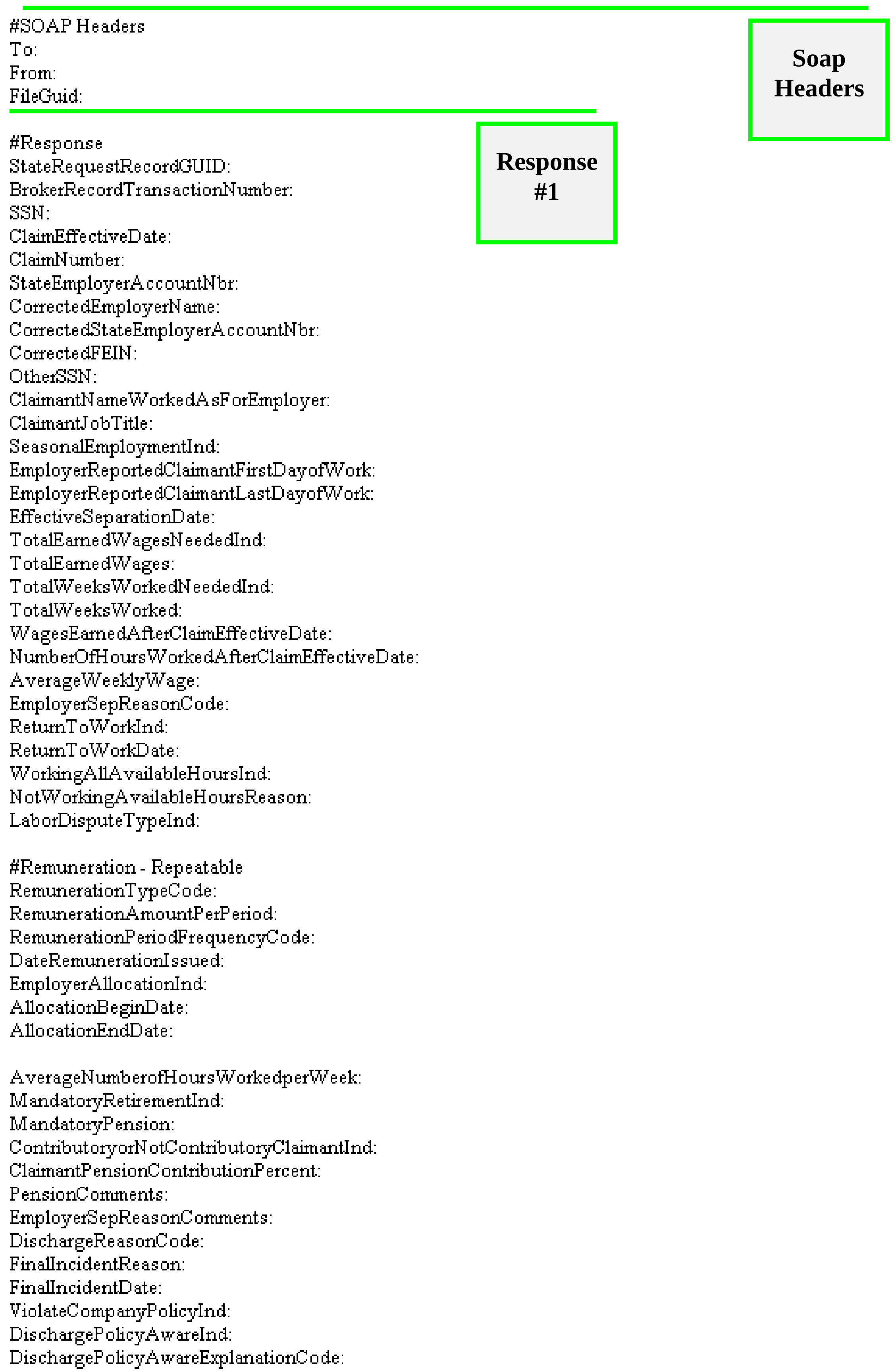
Input files are specified on the command line used to execute the Model Connector. See examples below.

7.2.1.4.3 Request Output

The SIDES Model Connector can provide requests in ASCII format, PDF, or both ASCII and PDF. In all cases, the requests are available in XML format. ASCII and PDF output files are specified in the runtime configuration parameters. The XML request output file is contained in the results Log file, whose path is specified in the runtime configuration parameters.

7.2.1.4.4 ASCII File Specification – Separation Information Post

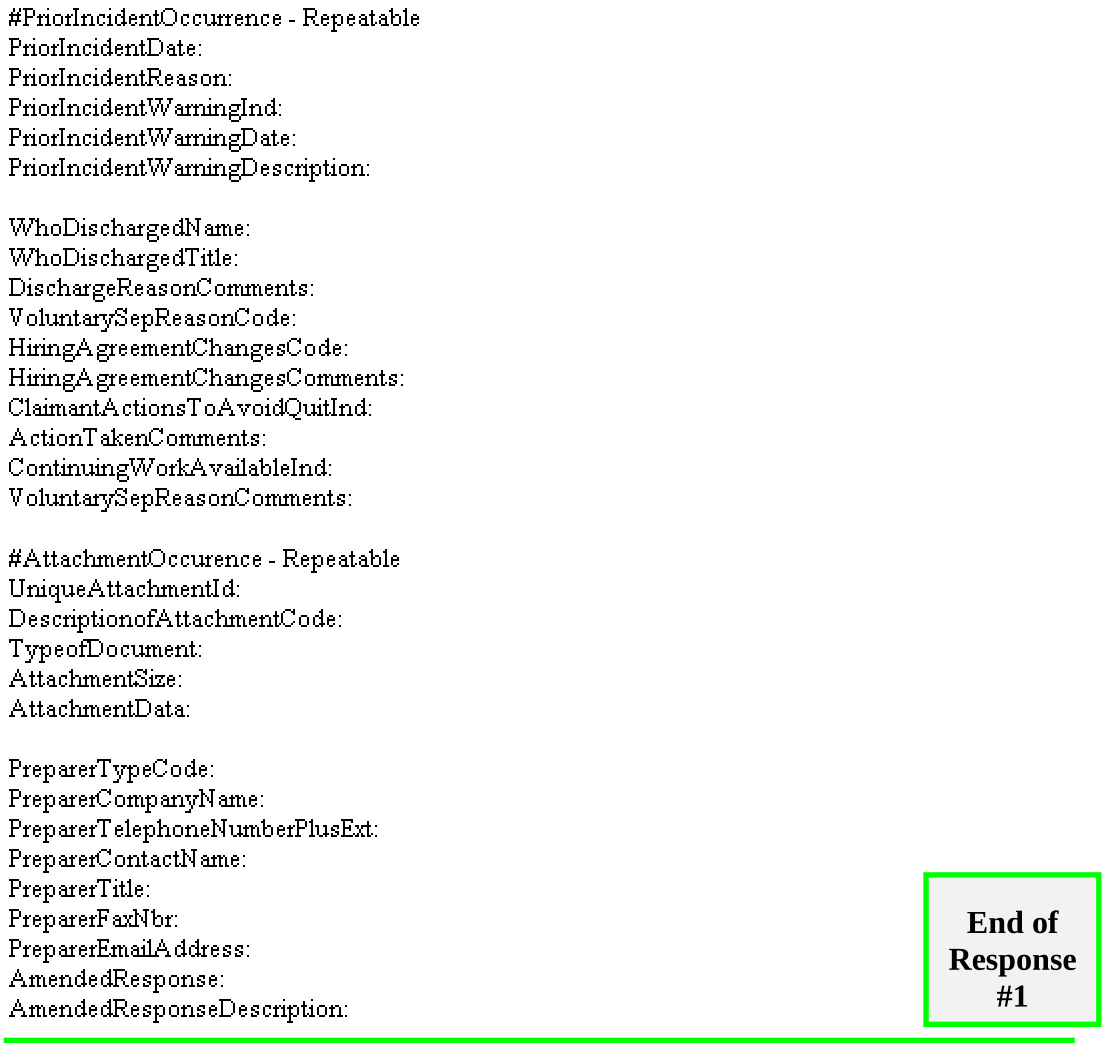
The ASCII file ingested by the Model Connector on the Post has two main sections. Section 1 (SOAP Headers) describes the SOAP Headers that must be placed on the SOAP message. The second section (Response) describes the actual response records. These are discussed further below. Any line in the ASCII specification that is empty or contains a # character as the first character is ignored. The # allows comments to be placed in the file.



Soap Headers

Response #1

```
#SOAP Headers
To:
From:
FileGuid:

#Response
StateRequestRecordGUID:
BrokerRecordTransactionNumber:
SSN:
ClaimEffectiveDate:
ClaimNumber:
StateEmployerAccountNbr:
CorrectedEmployerName:
CorrectedStateEmployerAccountNbr:
CorrectedFEIN:
OtherSSN:
ClaimantNameWorkedAsForEmployer:
ClaimantJobTitle:
SeasonalEmploymentInd:
EmployerReportedClaimantFirstDayofWork:
EmployerReportedClaimantLastDayofWork:
EffectiveSeparationDate:
TotalEarnedWagesNeededInd:
TotalEarnedWages:
TotalWeeksWorkedNeededInd:
TotalWeeksWorked:
WagesEarnedAfterClaimEffectiveDate:
NumberOfHoursWorkedAfterClaimEffectiveDate:
AverageWeeklyWage:
EmployerSepReasonCode:
ReturnToWorkInd:
ReturnToWorkDate:
WorkingAllAvailableHoursInd:
NotWorkingAvailableHoursReason:
LaborDisputeTypeInd:

#Remuneration - Repeatable
RemunerationTypeCode:
RemunerationAmountPerPeriod:
RemunerationPeriodFrequencyCode:
DateRemunerationIssued:
EmployerAllocationInd:
AllocationBeginDate:
AllocationEndDate:

AverageNumberofHoursWorkedperWeek:
MandatoryRetirementInd:
MandatoryPension:
ContributoryorNotContributoryClaimantInd:
ClaimantPensionContributionPercent:
PensionComments:
EmployerSepReasonComments:
DischargeReasonCode:
FinalIncidentReason:
FinalIncidentDate:
ViolateCompanyPolicyInd:
DischargePolicyAwareInd:
DischargePolicyAwareExplanationCode:
```



```
#PriorIncidentOccurrence - Repeatable
PriorIncidentDate:
PriorIncidentReason:
PriorIncidentWarningInd:
PriorIncidentWarningDate:
PriorIncidentWarningDescription:

WhoDischargedName:
WhoDischargedTitle:
DischargeReasonComments:
VoluntarySepReasonCode:
HiringAgreementChangesCode:
HiringAgreementChangesComments:
ClaimantActionsToAvoidQuitInd:
ActionTakenComments:
ContinuingWorkAvailableInd:
VoluntarySepReasonComments:

#AttachmentOccurence - Repeatable
UniqueAttachmentId:
DescriptionofAttachmentCode:
TypeofDocument:
AttachmentSize:
AttachmentData:

PreparerTypeCode:
PreparerCompanyName:
PreparerTelephoneNumberPlusExt:
PreparerContactName:
PreparerTitle:
PreparerFaxNbr:
PreparerEmailAddress:
AmendedResponse:
AmendedResponseDescription:
```

End of Response #1

The SOAP headers section of the ASCII file contains the routing information discussed in section 4.3-SOAP Custom Headers. It must contain the following information:

Table 38 - Employer Post to Broker

Header Element	Required	Definition	Example
To	Y	The Unique ID of the State to which the message is intended	UT
From	Y	The Unique ID of the employer/TPA where the message originated	BR000000003
FileGUID	Y	The employer-generated GUID applied to this message that can uniquely identify	A42A1FBDAC9549 AC7D8D3F45E404

Header Element	Required	Definition	Example
		this file Size is 32 hexadecimal digits	0319

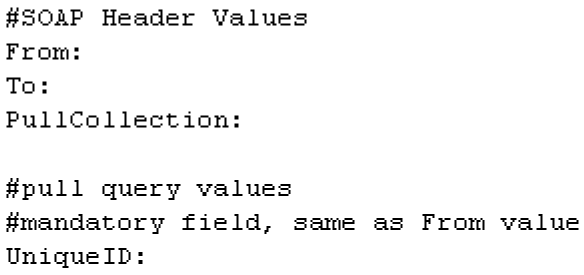
The Response section of the ASCII file contains the actual response record or records. There can be multiple responses records in a file. Each request record must start with the element name StateRequestRecordGUID. Within the record itself, there can also be repeatable attachment sections (up to 10 attachments according to the Separation Information specification), Prior Incidents or Remunerations. The repeatable section must contain all of the information on a particular section before moving on to the next repeatable element. Each line in the record contains the Data Element Name as described in the Implementation Guide followed by a colon (:) followed by the value given to that element name. The value must be contained all on a single line. If the data element value is null, it must not be in the ASCII file.

7.2.1.4.5 ASCII File Specification – Separation Information PULL

The ASCII file ingested by the Model Connector on the Pull has two main sections. Section 1 describes the SOAP Headers, which must be placed on the SOAP message. Section 2 describes the Pull Collection Query. These are discussed below. Any line that is empty or has a # character as the first character in it is ignored. The # allows comments to be placed in the file.

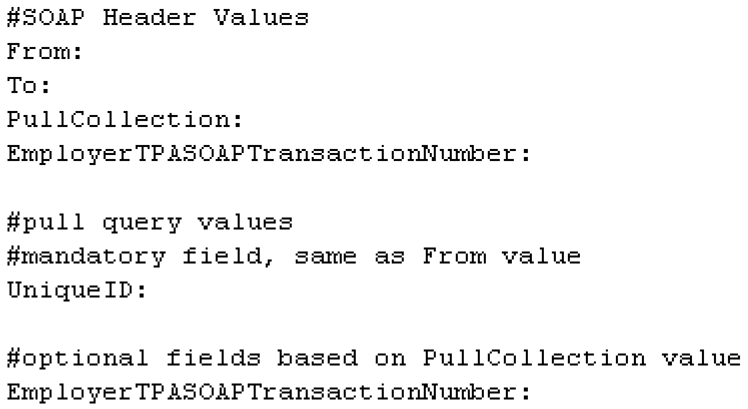
There are three PULL formats that will accomplish the same thing as their XML counterparts described in section 4.3.2-State Pull.

- Regular Pull



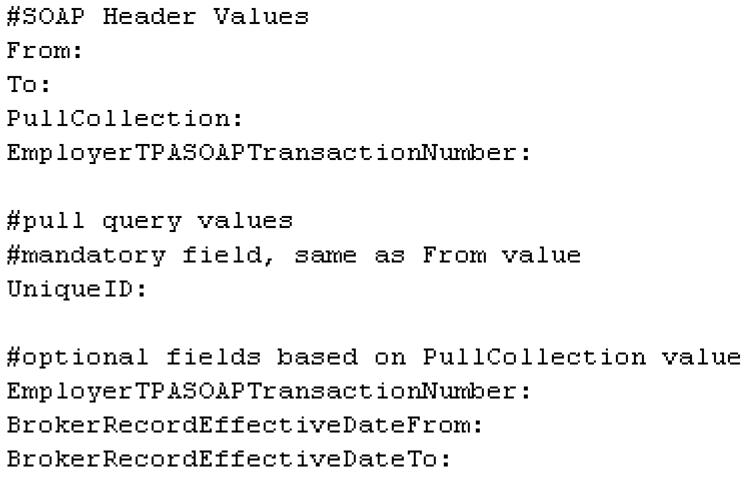
```
#SOAP Header Values
From:
To:
PullCollection:

#pull query values
#mandatory field, same as From value
UniqueID:
```

- Pull By EmployerTPA Soap Transaction Number



```
#SOAP Header Values
From:
To:
PullCollection:
EmployerTPASOAPTransactionNumber:

#pull query values
#mandatory field, same as From value
UniqueID:

#optional fields based on PullCollection value
EmployerTPASOAPTransactionNumber:
```

- Pull By Date



```
#SOAP Header Values
From:
To:
PullCollection:
EmployerTPASOAPTransactionNumber:

#pull query values
#mandatory field, same as From value
UniqueID:

#optional fields based on PullCollection value
EmployerTPASOAPTransactionNumber:
BrokerRecordEffectiveDateFrom:
BrokerRecordEffectiveDateTo:
```

If specified in the configuration file, the Employer Model Connector will return the requests in an ASCII format. The top of the file will contain the SOAP Headers in the response file.

Table 39 - Broker Response to Request (Regular Pull)

Header Element	Definition	Example
To	The Unique ID of the employer/TPA that requested the Pull	BR000000003
From	The Unique ID of the state from which these request records originated	UT
EmployerTPASOAPTransactionNumber	The unique number assigned to this file by the Broker	3565
MessageCode	The acknowledgement code applied to the	1

Header Element	Definition	Example
	message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	

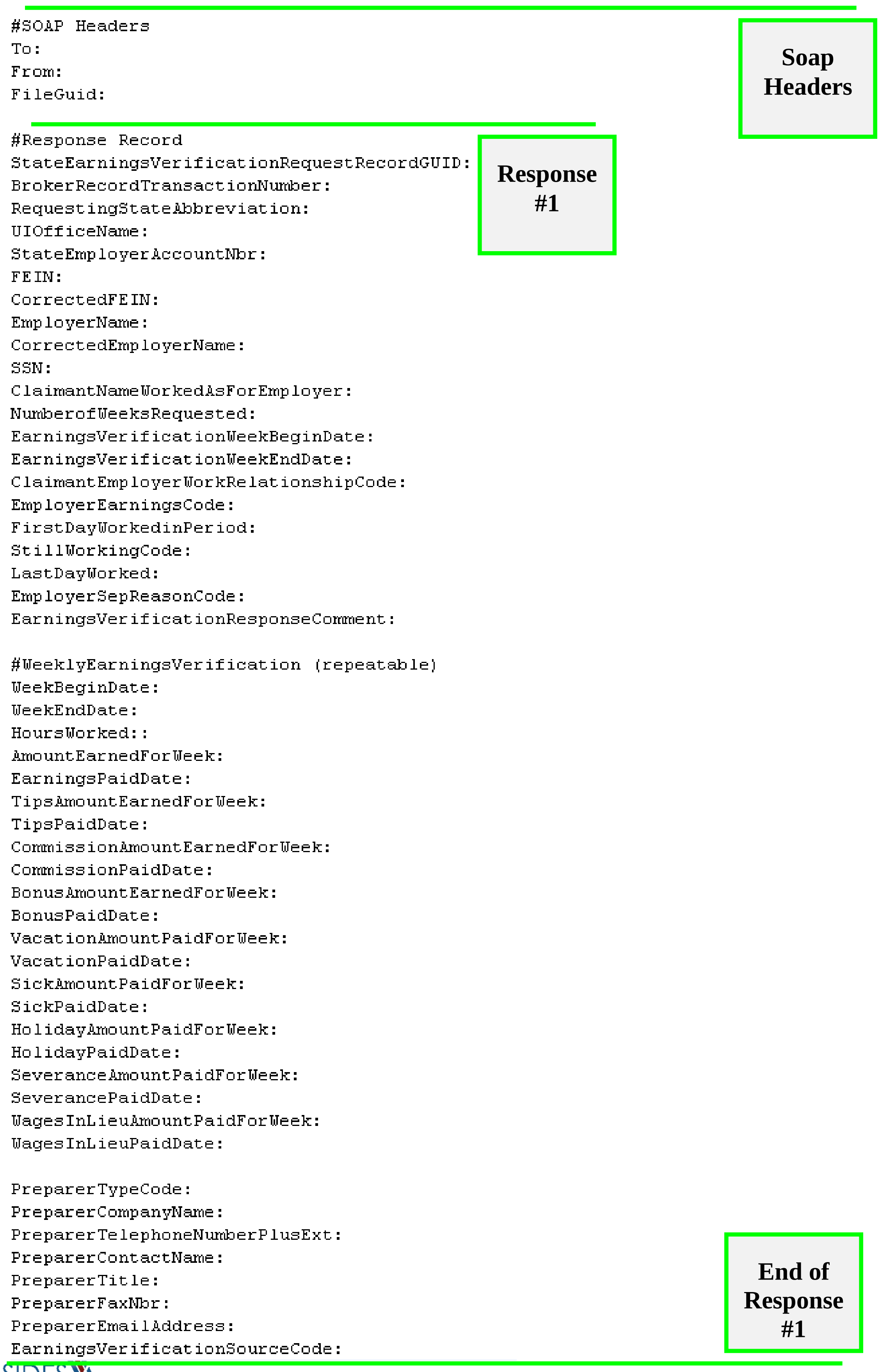
After the SOAP headers, the ASCII file will contain all the records returned in the pull call. The format will be the same as described in the post, where there is an element name followed by a colon (:) followed by the value being returned. A complete file specification containing all data elements is not provided as the returned file is dynamic, based upon the business rules. SIDES participants should rely on the standard format to ensure all response values are ingested by the back-end system. The following is an example Request file.

```
#Separation Request
StateRequestRecordGUID:ccc5915556584c3fad5ef6d21de9eb23
SSN:000195788
ClaimEffectiveDate:2010-07-11
StateEmployerAccountNbr:129054
EmployerName:H E BUTT GROCERY COMPANY
FEIN:740537175
TypeofEmployerCode:2
TypeofClaimCode:1
BenefitYearBeginDate:2010-07-11
RequestingStateAbbreviation:TX
UIOfficeName:TEXAS WORKFORCE COMMISSIO
UIOfficePhone:8888766107
UIOfficeFax:5123222815
ClaimantLastName:HUGHES
ClaimantFirstName:BRIAN
ClaimantMiddleInitial:K
ClaimantJobTitle:AVIATION FULLER
ClaimantReportedFirstDayofWork:2008-10-13
ClaimantReportedLastDayofWork:2010-06-29
WagesWeeksNeededCode:NA
ClaimantSepReasonCode:6

RequestDate:2010-08-18
ResponseDueDate:2010-09-01
FormNumber:610.0
BrokerRecordTransactionNumber:2041311
BrokerRecordEffectiveDate:2011-04-14T10:40:42-0400

#Separation Request
StateRequestRecordGUID:c755d30ed7dd4662bc0452e9050c00df
SSN:000989494
ClaimEffectiveDate:2008-09-28
ClaimNumber:0
StateEmployerAccountNbr:342424001
EmployerName:ELDORA ENTERPRISES LTD LIABILITY CO
```

```
FEIN:841173055
TypeofEmployerCode:1
TypeofClaimCode:1
BenefitYearBeginDate:2008-09-28
RequestingStateAbbreviation:CO
UIOfficeName:CO CDLE
UIOfficePhone:3033189055
UIOfficeFax:3033189014
ClaimantLastName:WHEELOCK
ClaimantFirstName:PHILIPPE
ClaimantMiddleInitial:M
ClaimantJobTitle:SKI PATROL
ClaimantReportedFirstDayofWork:2005-11-25
ClaimantReportedLastDayofWork:2008-04-10
WagesWeeksNeededCode:NA
ClaimantSepReasonCode:1

#Attachment occurence
UniqueAttachmentId:1
DescriptionofAttachmentCode:3
TypeofDocument:NOTICE AND REQUEST FOR SEPARATION INFO
ActionableAttachment:3
AttachmentSize:53104
AttachmentData:e1xydGYxXGFkZWZsYW5nMTAyNVxhbnNpXGFuc2ljcGcxMjUyXHVjMVxhZGVmZj
BcZGVmZjBccQ==

RequestDate:2008-09-28
ResponseDueDate:2008-10-13
FormNumber:UIB-290e
BrokerRecordTransactionNumber:2041294
BrokerRecordEffectiveDate:2011-04-13T17:31:52-0400
```

7.2.1.4.6 ASCII File Specification – Earnings Verification POST

The ASCII file ingested by the Model Connector on the Post has two main sections to it. Section 1 describes the SOAP Headers that must be placed on the SOAP message. Section 2 describes the actual response record(s). These are discussed below. Any line that is empty or has a # character as the first character in it is ignored. The # allows comments to be placed in the file.



Soap Headers
Response #1
End of Response #1

```
#SOAP Headers
To:
From:
FileGuid:

#Response Record
StateEarningsVerificationRequestRecordGUID:
BrokerRecordTransactionNumber:
RequestingStateAbbreviation:
UIOfficeName:
StateEmployerAccountNbr:
FEIN:
CorrectedFEIN:
EmployerName:
CorrectedEmployerName:
SSN:
ClaimantNameWorkedAsForEmployer:
NumberofWeeksRequested:
EarningsVerificationWeekBeginDate:
EarningsVerificationWeekEndDate:
ClaimantEmployerWorkRelationshipCode:
EmployerEarningsCode:
FirstDayWorkedinPeriod:
StillWorkingCode:
LastDayWorked:
EmployerSepReasonCode:
EarningsVerificationResponseComment:

#WeeklyEarningsVerification (repeatable)
WeekBeginDate:
WeekEndDate:
HoursWorked::
AmountEarnedForWeek:
EarningsPaidDate:
TipsAmountEarnedForWeek:
TipsPaidDate:
CommissionAmountEarnedForWeek:
CommissionPaidDate:
BonusAmountEarnedForWeek:
BonusPaidDate:
VacationAmountPaidForWeek:
VacationPaidDate:
SickAmountPaidForWeek:
SickPaidDate:
HolidayAmountPaidForWeek:
HolidayPaidDate:
SeveranceAmountPaidForWeek:
SeverancePaidDate:
WagesInLieuAmountPaidForWeek:
WagesInLieuPaidDate:

PreparerTypeCode:
PreparerCompanyName:
PreparerTelephoneNumberPlusExt:
PreparerContactName:
PreparerTitle:
PreparerFaxNbr:
PreparerEmailAddress:
EarningsVerificationSourceCode:
```

The first part of the ASCII file contains the SOAP headers. This is the routing information discussed in section 4.3-SOAP Custom Headers. It must contain the following information:

Table 40 - Employer Post to Broker

Header Element	Required	Definition	Example
To	Y	The Unique ID of the state to which the message is intended	UT
From	Y	The Unique ID of the employer/TPA where the message originated	BR000000003
FileGUID	Y	The employer-generated GUID applied to this message that can uniquely identify this file Size is 32 hexadecimal digits	A42A1FBDAC9549AC7D8D3F45E4040319

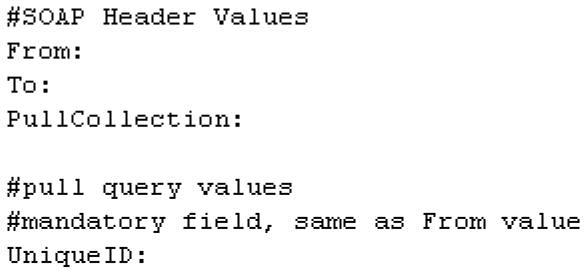
The second part of the ASCII file is the actual response record(s). There can be multiple response records in a file. Each response record must begin with the element name EmployerTPAEarningsVerificationRequestRecordGUID. Each line in the record contains the Data Element Name as described in the Implementation Guide followed by a colon (:) followed by the value given to that element name. The value must be contained all on a single line. If the data element value is null, it must not be in the ASCII file.

7.2.1.4.7 ASCII File Specification – Earnings Verification PULL

The ASCII file ingested by the Model Connector on the Pull has two main sections to it. The first section describes the SOAP Headers that must be placed on the SOAP message. The second section describes the Pull Collection Query. These are discussed below. Any line that is empty or has a # character as the first character in it is ignored. The # allows comments to be placed in the file.

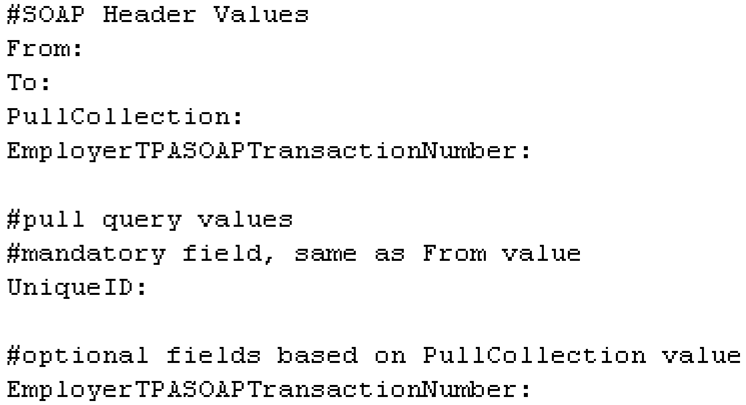
There are three PULL formats that will accomplish the same thing as their XML counterparts described in section 4.3.2-State Pull.

- Regular Pull



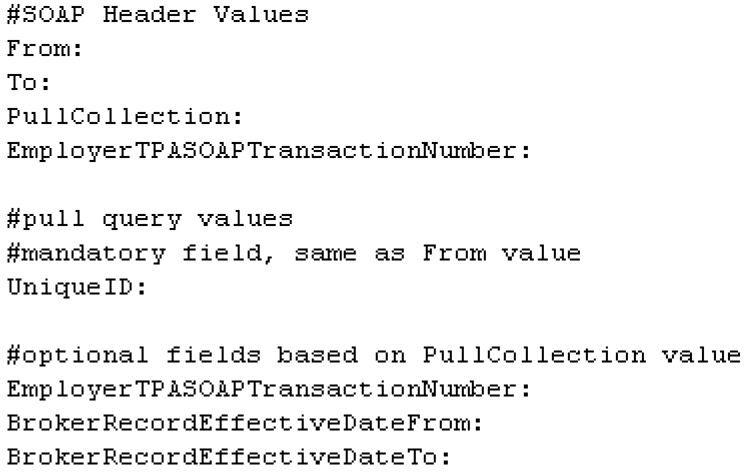
```
#SOAP Header Values
From:
To:
PullCollection:

#pull query values
#mandatory field, same as From value
UniqueID:
```

- Pull By EmployerTPA Soap Transaction Number



```
#SOAP Header Values
From:
To:
PullCollection:
EmployerTPASOAPTransactionNumber:

#pull query values
#mandatory field, same as From value
UniqueID:

#optional fields based on PullCollection value
EmployerTPASOAPTransactionNumber:
```

- Pull By Date



```
#SOAP Header Values
From:
To:
PullCollection:
EmployerTPASOAPTransactionNumber:

#pull query values
#mandatory field, same as From value
UniqueID:

#optional fields based on PullCollection value
EmployerTPASOAPTransactionNumber:
BrokerRecordEffectiveDateFrom:
BrokerRecordEffectiveDateTo:
```

If specified in the configuration file, the Employer Model Connector will return the request in an ASCII format. The top of the file will contain the SOAP Headers in the response file.

Table 41 - Broker Response to Request (Regular Pull)

Header Element	Definition	Example
To	The Unique ID of the employer/TPA that	BR000000001

Header Element	Definition	Example
	requested the Pull	
From	The Unique ID of the state from which these request records originated	UT
EmployerTPASOAPTransactionNumber	The unique number assigned to this file by the Broker	3565
MessageCode	The acknowledgement code applied to the message that indicates success or failure of the entire transmission. See 4.2.5 for further information on Message Codes. Size is one digit	1

After the SOAP headers, the ASCII file will contain all the records returned in the pull call. The format will be the same as described in the post, where there is an element name followed by a colon (:) followed by the value being returned. A complete file specification containing all data elements is not provided as the returned file is dynamic, based upon the business rules and request indicator values. SIDES participants should rely on the standard format to ensure all response values are ingested by the back-end system. The following is an example Request file.

```
#Request Payload
#SOAP Headers
To:BR999999999
From:ST
EmployerTPASOAPTransactionNumber:9551
MessageCode:1


#Earnings Verification Request
StateEarningsVerificationRequestRecordGUID:c755d30ed7dd4662bc0452e9050c00cd
RequestingStateAbbreviation:ST
UIOfficeName:UI Office of ST
UIOfficePhone:2105551212
UIOfficeFax:2105551313
UIOfficeEmailAddress:test@nowhere.com
StateEmployerAccountNbr:0123456789
FEIN:999999999
EmployerName:Test Employer
SSN:000989496
ClaimantLastName:Doe
ClaimantFirstName:John
ClaimantMiddleInitial:M
ClaimantSuffix:Jr
```

```
NumberofWeeksRequested:5
EarningsVerificationWeekBeginDate:2010-01-03
EarningsVerificationWeekEndDate:2010-02-06
EarningsVerificationComments:Test of Earnings Verification Comments field.
RequestDate:2010-09-07
EarningsStatusCode:3
TipsStatusCode:1
CommissionStatusCode:1
BonusStatusCode:3
VacationStatusCode:2
SickLeaveStatusCode:3
HolidayStatusCode:2
SeveranceStatusCode:2
WagesInLieuStatusCode:2
EarningsVerificationResponseCommentIndicator:1
ResponseDueDate:2012-12-07
EarningsVerificationSourceCode:9
BrokerRecordTransactionNumber:7850
BrokerRecordEffectiveDate:2011-04-22T12:46:49-0400

#SOAP Headers
To:BR999999999
From:Broker
EmployerTPASOAPTransactionNumber:9229
MessageCode:2
```

7.2.1.5 Log Files – POST

7.2.1.5.1 DEBUG log file

This log file is the main debugging log file for a given run. It contains all debug output logged in the system during that run. If the system were to fail unexpectedly, the log file will contain the most up to date status and will most likely indicate where the system failed. It also includes all the data that is written to the other log files.

The log file is placed into the directory specified by the DebugLogFilePath configuration parameter. The naming scheme is as follows:

```
POST_{SI|EV}_DEBUG_date_time.log

For example:

POST_EV_DEBUG_2011-04-08_15-23-18-292.log
```

7.2.1.5.2 BRPT log file

This log file shows the results from the call to the BRPT on the response files submitted for a given run. It will indicate all the records that had a problem in them and were thus stripped for the response file being sent. It is the responsibility of the employer or TPA to correct these errors and retransmit these responses to the **Central Broker**.

An example of the contents of a BRPT log file follows below:

```
#Failed Records
Record GUID Failure:00000000000000000000000060000180
Number of errors detected:1
#Errors
Error Number:1
Error Code:248
Error Message:Business Rule violation - There must be a value (Character -
Size 60) for PreparerCompanyName if PreparerTypeCode = T for Third Party
Administrator
Record GUID Failure:00000000000000000000000060000181
Number of errors detected:1
#Errors
Error Number:1
Error Code:248
Error Message:Business Rule violation - There must be a value (Character -
Size 60) for PreparerCompanyName if PreparerTypeCode = T for Third Party
Administrator
```

The log file is placed into the directory specified by the BrptLogFilePath configuration parameter. The naming scheme is as follows:

```
POST_{SI|EV}_BRPT_{to}_{from}_date_time_guid.log

For example:

POST_EV_BRPT_ST_BR999999999_2011-04-08_15-23-18-
292_01234567890123456789012345678901.log
```

7.2.1.5.3 RESULTS log file

This log file shows the acknowledgement from the **Central Broker** for a given run.

An example illustrating the contents of a RESULTS log file follows below:

```
#SOAP Headers in Acknowledgement
To:BR999999999
EmployerTPAResponseFileGUID:577E92EE2CD5EE8C44B90A5A581B36F4
From:Broker
MessageCode:1
#Acknowledgement
File GUID:577E92EE2CD5EE8C44B90A5A581B36F4
Number of response Records Received:1
Number of response Records Received in Error:0
Date Started Received:2011-04-13T09:38:20.554-04:00
Date Finished Receiving:2011-04-13T09:38:20.870-04:00
```

The log file is placed into the directory specified by the ResultsLogFilePath configuration parameter. The naming scheme is as follows:

```
POST_{SI|EV}_RESULTS_{to}_{from}_date_time_guid.log
```

```
For example:

POST_SI_RESULTS_BR999999999_ST_2011-04-08_15-23-18-
292_01234567890123456789012345678901.log
```

7.2.1.6 Log Files – PULL

7.2.1.6.1 DEBUG log file

This is the main debugging log file for the whole application for a given run. It contains all debug output logged in the system during the run. If the system were to fail unexpectedly, this log file will contain the most up to date status and will most likely indicate where the system failed. It also includes all the data written to the other log files.

The log file is placed into the directory specified by the DebugLogFilePath configuration parameter. The naming scheme is as follows:

```
PULL_{SI|EV}_DEBUG_date_time.log

For example:

PULL_EV_DEBUG_2011-04-08_15-46-12-035.log
```

7.2.1.6.2 BRPT log file

This log file shows the results from the call to the BRPT on the request files returned by the **Central Broker** for a given run. If there are any records in this file, then the Employer Model Connector will return a Message Code of 2 back to the **Central Broker** indicating a failure. This file will then be pulled again on its next Pull call.

Here is an example of the contents of a BRPT log file:

```
#Failed Records
Record GUID Failure:30000000000000000000000000004000
Number of errors detected: 1
#Errors
Error Number:1
Error Code:101
Error Message:XSD validation violation
```

The log file is placed into the directory specified by the BrptLogFilePath configuration parameter. The naming scheme is as follows:

```
PULL_{SI|EV}_BRPT_{to}_{from}_date_time_PullCollection.log

For example:
```

```
PULL_EV_BRPT_Broker_BR999999999_2011-04-08_15-46-12-035_1.log
```

7.2.1.6.3 RESULTS log file

This log file shows the results of the Pull call from the **Central Broker** for a given run. It will contain the Employer/TPA Separation Requests in XML format. If the pullAllFiles config file parameter is set to true, then this file will contain all of the SOAP Headers and Response Payloads the Model Connector received from the **Central Broker**.

Here is an example of the contents of a RESULTS log file.

```
#SOAP Headers
To:BR999999999
From:ST
EmployerTPASOAPTransactionNumber: 143650
MessageCode:1

#Request Payload
<EmployerTPASeparationRequestCollection xmlns:ns2="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-ws        -utility-1.0.xsd"
xmlns="https:// REDACTED /schemas">
    <EmployerTPASeparationRequest>

<StateRequestRecordGUID>c755d30ed7dd4662bc0452e9050c00df</StateRequestRecordG
UID>
        <SSN>000989494</SSN>
        <ClaimEffectiveDate>2008-09-28</ClaimEffectiveDate>
        <ClaimNumber>0</ClaimNumber>
        <StateEmployerAccountNbr>342424001</StateEmployerAccountNbr>
        <EmployerName>ELDORA ENTERPRISES LTD LIABILITY CO</EmployerName>
        <FEIN>841173055</FEIN>
        <TypeofEmployerCode>1</TypeofEmployerCode>
        <TypeofClaimCode>1</TypeofClaimCode>
        <BenefitYearBeginDate>2008-09-28</BenefitYearBeginDate>
        <RequestingStateAbbreviation>CO</RequestingStateAbbreviation>
        <UIOfficeName>CO CDLE</UIOfficeName>
        <UIOfficePhone>3033189055</UIOfficePhone>
        <UIOfficeFax>3033189014</UIOfficeFax>
        <ClaimantLastName>WHEELOCK</ClaimantLastName>
        <ClaimantFirstName>PHILIPPE</ClaimantFirstName>
        <ClaimantMiddleInitial>M</ClaimantMiddleInitial>
        <ClaimantJobTitle>SKI PATROL</ClaimantJobTitle>
        <ClaimantReportedFirstDayofWork>2005-11-
25</ClaimantReportedFirstDayofWork>
        <ClaimantReportedLastDayofWork>2008-04-
10</ClaimantReportedLastDayofWork>
        <WagesWeeksNeededCode>NA</WagesWeeksNeededCode>
        <ClaimantSepReasonCode>1</ClaimantSepReasonCode>
        <AttachmentOccurrence>
            <UniqueAttachmentId>1</UniqueAttachmentId>
            <DescriptionofAttachmentCode>2</DescriptionofAttachmentCode>
            <TypeofDocument>NOTICE AND REQUEST FOR SEPARATION
INFO</TypeofDocument>
            <ActionableAttachment>3</ActionableAttachment>
            <AttachmentSize>53104</AttachmentSize>

<AttachmentData>e1xydGYxXGFkZWZsYW5nMTAyNVxhbnNpXGFuc2ljcGcxMjUyXHVjMVxhZGVmZ
jBcZGVmZjBccQ==</AttachmentData>
        </AttachmentOccurrence>
        <RequestDate>2008-09-28</RequestDate>
        <ResponseDueDate>2008-10-13</ResponseDueDate>
        <FormNumber>UIB-290e</FormNumber>

<BrokerRecordTransactionNumber>2041294</BrokerRecordTransactionNumber>
        <BrokerRecordEffectiveDate>2011-04-13T17:31:52.000-
04:00</BrokerRecordEffectiveDate>
    </EmployerTPASeparationRequest>
    <EmployerTPASeparationRequest>

<StateRequestRecordGUID>c755d30ed7dd4662bc0452e9050c00df</StateRequestRecordG
UID>
        <SSN>000989494</SSN>
        <ClaimEffectiveDate>2008-09-28</ClaimEffectiveDate>
        <ClaimNumber>0</ClaimNumber>
        <StateEmployerAccountNbr>342424001</StateEmployerAccountNbr>
        <EmployerName>ELDORA ENTERPRISES LTD LIABILITY CO</EmployerName>
        <FEIN>841173055</FEIN>
        <TypeofEmployerCode>1</TypeofEmployerCode>
        <TypeofClaimCode>1</TypeofClaimCode>
        <BenefitYearBeginDate>2008-09-28</BenefitYearBeginDate>
        <RequestingStateAbbreviation>CO</RequestingStateAbbreviation>
        <UIOfficeName>CO CDLE</UIOfficeName>
        <UIOfficePhone>3033189055</UIOfficePhone>
        <UIOfficeFax>3033189014</UIOfficeFax>
        <ClaimantLastName>WHEELOCK</ClaimantLastName>
        <ClaimantFirstName>PHILIPPE</ClaimantFirstName>
        <ClaimantMiddleInitial>M</ClaimantMiddleInitial>
        <ClaimantJobTitle>SKI PATROL</ClaimantJobTitle>
        <ClaimantReportedFirstDayofWork>2005-11-
25</ClaimantReportedFirstDayofWork>
        <ClaimantReportedLastDayofWork>2008-04-
10</ClaimantReportedLastDayofWork>
        <WagesWeeksNeededCode>NA</WagesWeeksNeededCode>
        <ClaimantSepReasonCode>1</ClaimantSepReasonCode>
        <AttachmentOccurrence>
            <UniqueAttachmentId>1</UniqueAttachmentId>
            <DescriptionofAttachmentCode>3</DescriptionofAttachmentCode>
            <TypeofDocument>NOTICE AND REQUEST FOR SEPARATION
INFO</TypeofDocument>
            <ActionableAttachment>3</ActionableAttachment>
            <AttachmentSize>53104</AttachmentSize>

<AttachmentData>e1xydGYxXGFkZWZsYW5nMTAyNVxhbnNpXGFuc2ljcGcxMjUyXHVjMVxhZGVmZ
jBcZGVmZjBccQ==</AttachmentData>
        </AttachmentOccurrence>
        <RequestDate>2008-09-28</RequestDate>
        <ResponseDueDate>2008-10-13</ResponseDueDate>
        <FormNumber>UIB-290e</FormNumber>

<BrokerRecordTransactionNumber>2041333</BrokerRecordTransactionNumber>
        <BrokerRecordEffectiveDate>2011-04-20T16:39:43.000-
04:00</BrokerRecordEffectiveDate>
    </EmployerTPASeparationRequest>
    <EmployerTPASeparationRequest>

<StateRequestRecordGUID>aee086161ef8499092f9f260154ea243</StateRequestRecordG
UID>
        <SSN>334620158</SSN>
        <ClaimEffectiveDate>2010-07-04</ClaimEffectiveDate>
        <StateEmployerAccountNbr>20941456</StateEmployerAccountNbr>
        <EmployerName>IKON OFFICE SOLUTIONS INC</EmployerName>
        <FEIN>230334400</FEIN>
        <TypeofEmployerCode>5</TypeofEmployerCode>
        <TypeofClaimCode>1</TypeofClaimCode>
        <BenefitYearBeginDate>2010-07-04</BenefitYearBeginDate>
        <RequestingStateAbbreviation>TX</RequestingStateAbbreviation>
        <UIOfficeName>TEXAS WORKFORCE COMMISSIO</UIOfficeName>
        <UIOfficePhone>8886578749</UIOfficePhone>
        <UIOfficeFax>5123222875</UIOfficeFax>
        <ClaimantLastName>ORNEDO</ClaimantLastName>
        <ClaimantFirstName>LINA</ClaimantFirstName>
        <ClaimantMiddleInitial>B</ClaimantMiddleInitial>
        <ClaimantJobTitle>ACCOUNTANT</ClaimantJobTitle>
        <ClaimantReportedFirstDayofWork>2008-08-
17</ClaimantReportedFirstDayofWork>
        <ClaimantReportedLastDayofWork>2010-07-
03</ClaimantReportedLastDayofWork>
        <WagesWeeksNeededCode>NA</WagesWeeksNeededCode>
        <ClaimantSepReasonCode>2</ClaimantSepReasonCode>
        <RequestDate>2010-07-07</RequestDate>
        <ResponseDueDate>2010-07-21</ResponseDueDate>
        <FormNumber>610.0</FormNumber>

<BrokerRecordTransactionNumber>2041315</BrokerRecordTransactionNumber>
        <BrokerRecordEffectiveDate>2011-04-14T10:46:23.000-
04:00</BrokerRecordEffectiveDate>
    </EmployerTPASeparationRequest>
</EmployerTPASeparationRequestCollection>

#SOAP Headers
To:BR999999999
From:Broker
EmployerTPASOAPTransactionNumber:143652
MessageCode:2

#Request Payload
<EmployerTPASeparationRequestCollection xmlns:ns2="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-ws        -utility-1.0.xsd"
xmlns="https:// REDACTED /schemas"/>
```

The log file is placed into the directory specified by the ResultsLogFilePath configuration parameter. The naming scheme is as follows:

```
PULL_{SI|EV}_RESULTS_{to}_{from}_date_time_PullCollection.log
```

```
For example:
```

```
PULL_SI_RESULTS_Broker_ST_2011-04-08_15-46-12-035_1.log
```

7.2.2 Model Connector – Spring

7.2.2.1 Spring-WS Model Connector

7.2.2.1.1 Model Connector Core Components

This Model Connector was developed using Java and the Spring framework.

Within the Model Connector, the **REDACTED** support is delegated by Spring-WS to Apache Wss4j.

The Model Connector also uses JAXB2 library to marshall/unmarshall XML to/from Java beans.

The Model Connector was developed on JDK 1.5 (Java 5), but should also run on JDK 1.4 and Java 6.

The following main libraries are used:

• Spring-2.5.6	(Core Spring library)
• Spring-ws-1.5.8	(Spring Web Services library)
• Apache Wss4j-1.5.8	(**REDACTED** provider)
• Stax-api-1.0.1	(Streaming API for XML)
• JAXB2 2.1.7	(JAXB2 marshaller/unmarshaller)

For convenience, this sample includes all necessary Eclipse project config files and can be imported into an existing Eclipse IDE workspace. Eclipse 3.5 (Galileo) or later is required.

7.2.2.1.2 Sample Folders and Files

Root folder: sides-state-client

- ./build.xml

  Ant build file (requires Apache Ant 1.7.1 or later)
  - Run "ant build" to compile
  - Run "ant run-post" to execute sample State Post ws call
  - Run "ant run-pull" to execute sample State Pull/State Pull Acknowledgement ws calls

- ./run-post.*

  Unix/Windows shell scripts to run sample State Post Model Connector

    - build sample with "ant build" first

- ./run-pull.*

  Unix/Windows shell scripts to run sample State Pull Model Connector
    - build sample with "ant build" first

- ./src

  Contains:
    - Java source code
    - The Spring config xml file (state-ws-emulator-config.xml)
    - Log4j config file (log4j.properties)
    - Sample Java

- ./lib
  Contains required library jar files. All libraries used are open-source Apache LGPL-style libraries which can be freely distributed.

- ./schemas –

  Contains UI **SIDES** XML schema files and State WSDL file

- ./data

  Contains sample payload xml data files for State Post (StateSIPost.xml) and State Pull Query (StateSIPullQuery.xml)

- ./bin

  Build destination folder for compiled Java class files.

7.2.2.1.3 RunTime Configuration

The Model Connector has runtime configuration parameters that allow the state to setup its connector. The configuration is specified in the Spring config xml file. The bean that specifies these parameters is the configParams bean. All Java Application Model Connector classes use the same Spring configuration file, ***state-ws-emulator-config.xml***.

Table 42 - ConfigParam options

Parameter Name	**Applies To**	**Definition**
debugLogFilePath	Post and Pull	The fully qualified location of the debug log file; it contains all the information in all the log files plus

Parameter Name	Applies To	Definition
		detailed information on the state of the Model Connectors workings.
resultsLogFilePath	Post and Pull	The fully qualified location of the results log file; it contains all the information with the results from the call to the Broker.
brptLogFilePath	Post and Pull	The fully qualified location of the brpt log file; it contains all the information with the results from the BRPT.
pinLogFilePath	Post	The fully qualified location of the pin log file; it contains all the information on the new pin created if the *createPin* config parameter is set to true
pdfFilePath	Pull	The fully qualified location of the PDF file and attachments; it will contain all the response received in PDF form with all of the attachments decoded and stored in the same directory
writeResponsesAsPDF	Pull	A boolean value that is "true" if the responses should be printed out as the PDF and "false" otherwise.
responseFlatFilePath	Pull	The fully qualified location of the flat file containing the Response information; it will contain all the responses received in flat file format with all of the attachments still encoded
writeResponsesAsFlatFile	Pull	A boolean value that is "true" if the responses should be written in the flat file format and "false" otherwise.
createPin	Post	A boolean value that is "true" if the system is directed to create the PIN for the request and "false" otherwise.
pullAllFiles	Pull	A boolean value that is "true" if the connector wants the system to pull all files until the message code = 2 and "false" if the connector wants to make the call repeatedly (so as to allow the connector more control).

7.2.2.1.4 Web Services Configuration

The URL of the **SIDES** Broker Web services and **REDACTED** configuration is specified in the Spring config xml file. All Java Application Model Connector classes use the same Spring configuration file, ***state-ws-emulator-config.xml***.

These are some important spring config file parameters:

The sample Model Connector uses **REDACTED** which is associated with the test-state endpoint (Connector Name: State Test; Unique Id: ST) on the test **SIDES** Central Broker deployment. To change the **REDACTED** used by Broker to **REDACTED** State's WS requests:

1. Import the new **REDACTED**
   - See the article at **REDACTED** for helpful hints
2. Update **REDACTED**
3. Send the updated information on the **REDACTED** to the Broker.

7.2.2.1.5 Execution

This Model Connector contains 4 top-level Java classes with a main() method:

- StatePostClient
- StatePostClientDataFile
- StatePullClient
- StatePullClientDataFile

7.2.2.1.5.1 StatePostClient/StatePullClient

These files implement the State Post and the State Pull/Pull Acknowledgment web service calls respectively.

The StatePostClient and StatePullClient use JAXB2 library for manipulating XML elements as Java beans.

The StatePostClient/StatePullClient Model Connectors read message payload content from XML files in the designated folder, send it to the test **SIDES** Broker Web services URL, and log the **Central Broker's** response to the console.

The State Post Model Connector expects five or seven command line arguments:

Table 43 – Spring State Post Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is

Model Connector Argument	Definition
	destined for: SI – Separation Information EV – Earnings Verification
"FROM" SOAP header	This is the unique id of the State that is sending the file.
"TO" SOAP header	This is the unique id of the Employer/TPA that the file is destined for.
"StateRequestFileGUID" SOAP header	This is the State Request File GUID.
The payload XML source file	The XML file that contains the payload for the call.
SEIN SOAP header (optional – StatePostClient only, Separation Information Only)	SEIN value if sending to **SEW** employer/TPA
PIN SOAP header (optional – StatePostClient only)	PIN value if sending to **SEW** employer/TPA

To execute the Model Connector from the command line, type:

- *java org.uisides.client.state.StatePostClient SI|EV FROM TO StateRequestFileGUID Payload_XML_File_Name [SEIN PIN]*

Sample Model Connector arguments are:

- *java org.uisides.client.state.StatePostClient SI ST BR999999999 12345678901234567890123456789012 data/StateSIPost.xml*

where:

- SI is the exchange the file is destined for
- ST is the State Test unique id
- BR999999999 is the unique id for EmployerTest UI **SIDES** endpoint
- 12345678901234567890123456789012 is a test StateRequestFileGUID.

The State Pull Model Connector expects five or six command line arguments:

Table 44 – Spring State Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
"FROM" SOAP header	This is the unique id of the State that is sending the file.
"TO" SOAP header	This must be “Broker.”
“PullCollection” SOAP header	1 for regular pull, 2 for re-pull by

Model Connector Argument	Definition
	transmission number, 3 for re-pull by date range
The payload XML source file	The XML file that contains the payload for the call.
StateSOAPTransactionNumber (optional)	The value of the “StateSOAPTransactionNumber” SOAP header. Only required if the pullCollection parameter is 2 (re-pull by transaction number) or 3 (re-pull by date range)

To execute the Model Connector from the command line, type:

- *java org.uisides.client.state.StatePullClient SI|EV FROM **Broker** PullCollection Payload_XML_File_Name [StateSOAPTransactionNumber]*

Sample Model Connector arguments are:

- *java org.uisides.client.state.StatePullClient SI ST Broker 1 data/StateSIPullQuery.xml*

where:

- SI is the exchange the file is destined for
- ST is the State Test unique id
- 1 is the pull collection for a regular pull

7.2.2.1.5.2 StatePostClientDataFile/StatePullClientDataFile

These files implement the State Post and the State Pull/Pull Acknowledgment Web service calls respectively that read ASCII files. See Figure 1 and Figure 2.

The StatePostClientDateFile/StatePullClientDataFile Model Connectors read message payload content from ASCII flat files in the designated folder, send it to the test SIDES Broker Web services URL, and log Broker's response to the console.

The State Post Data File Model Connector expects two command line arguments:

Table 45 – Spring State Post Data File Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Data File	This is the fully qualified path and name of

	the data file that contains the flat file structure of the Request(s).

To execute the Model Connector from the command line, type:

- *java org.uisides.client.state.StatePostClientDataFile SI|EV Data_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.state.StatePostClientDataFile SI data/StateSIPost.txt*

  where:

  - SI is the exchange the file is destined for

Example State Request File

```
#SOAP Header Values
To:BR999999999
From:ST
FileGuid:01234567890123456789012345678901

StateRequestRecordGUID:c755d30ed7dd4662bc0452e9050c00df
SSN:000989494
ClaimEffectiveDate:2008-09-28
ClaimNumber:0
StateEmployerAccountNbr:342424001
EmployerName:ELDORA ENTERPRISES LTD LIABILITY CO
FEIN:841173055
TypeofEmployerCode:1
TypeofClaimCode:1
BenefitYearBeginDate:2008-09-28
RequestingStateAbbreviation:CO
UIOfficeName:CO CDLE
UIOfficePhone:3033189055
UIOfficeFax:3033189014
ClaimantLastName:WHEELOCK
ClaimantFirstName:PHILIPPE
ClaimantMiddleInitial:M
ClaimantJobTitle:SKI PATROL
ClaimantReportedFirstDayofWork:2005-11-25
ClaimantReportedLastDayofWork:2008-04-10
WagesWeeksNeededCode:NA
ClaimantSepReasonCode:1

UniqueAttachmentId:01
DescriptionofAttachmentCode:1
TypeofDocument:test-file.txt
ActionableAttachment:3
AttachmentSize:2000
```

AttachmentData:QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQU
FBDQpCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCDQp
DQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDDQpERERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERREQNCkVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUUNCkZGRkZGRkZGRkZGRkZGRk
ZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkYNCkdHR0dHR0dHR0dHR0dHR0dHR0d
HR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0cNCkhISEhISEhISEhISEhISEhISEhISEhISEhI
SEhISEhISEhISEhISEhISEhISEgNCklJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJS
UlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJDQpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSk
pKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSg0KS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0t
LS0tLS0tLS0tLS0tLS0tLSw0KTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExM
TExMTExMTExMTEwNCk1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU0NCk5OT
k5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk4NCk9PT09PT09PT0
9PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT08NClBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQDQpRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRDQpSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSU
lJSUlJSUlJSUlJSUlJSUlJSUlJSUg0KU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1
NTU1NTU1NTU1NTU1NTU1MNClRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUDQpBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUENCkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkINC
kNDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0MNCkRERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERA0KRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRQ0KRkZGRkZGRkZGRkZG
RkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRg0KR0dHR0dHR0dHR0dHR0dHR0dHR
0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHRw0KSEhISEhISEhISEhISEhISEhISEhISEhISEhISE
hISEhISEhISEhISEhISEhISEhISEhISA0KSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUl
JSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUkNCkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpK
SkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKDQpLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS
0tLS0tLS0tLS0tLS0tLS0tLDQpMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE
xMTExMTExMTExMTA0KTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTQ0KTk5
OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTg0KT09PT09PT09P
T09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PTw0KUFBQUFBQUFBQUFBQUFBQU
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFANClFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVENClJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJ
SUlJSUlJSUlJSUlJSUlJSUlJSUlJSDQpTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NT
U1NTU1NTU1NTU1NTU1NTUw0KVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFQNCg==

UniqueAttachmentId:02
DescriptionofAttachmentCode:1
TypeofDocument:test-file2.txt
ActionableAttachment:3
AttachmentSize:2000

AttachmentData:QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQU
FBDQpCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCDQp
DQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDDQpERERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERREQNCkVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUUNCkZGRkZGRkZGRkZGRkZGRk
ZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkYNCkdHR0dHR0dHR0dHR0dHR0dHR0d
HR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0cNCkhISEhISEhISEhISEhISEhISEhISEhISEhI
SEhISEhISEhISEhISEhISEhISEgNCklJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJS
UlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJDQpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSk
pKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSg0KS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0t
LS0tLS0tLS0tLS0tLS0tLSw0KTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExM
TExMTExMTExMTEwNCk1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU0NCk5OT
k5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk4NCk9PT09PT09PT0

```
9PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT08NClBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQDQpRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRDQpSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSU
lJSUlJSUlJSUlJSUlJSUlJSUlJSUg0KU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1
NTU1NTU1NTU1NTU1NTU1NTU1MNClRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUDQpBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUENCkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkINC
kNDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0MNCkRERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERA0KRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRQ0KRkZGRkZGRkZGRkZGRkZG
RkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRg0KR0dHR0dHR0dHR0dHR0dHR0dHR
0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHRw0KSEhISEhISEhISEhISEhISEhISEhISEhISE
hISEhISEhISEhISEhISEhISEhISA0KSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUl
JSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUkNCkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpK
SkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKDQpLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS
0tLS0tLS0tLS0tLS0tLS0tLDQpMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE
xMTExMTExMTExMTA0KTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTQ0KTk5
OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTg0KT09PT09PT09P
T09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PTw0KUFBQUFBQUFBQUFBQUFBQUFBQU
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFANClFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVENClJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJ
SUlJSUlJSUlJSUlJSUlJSUlJSUlJSDQpTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NT
U1NTU1NTU1NTU1NTU1NTUw0KVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFQNCg==
```

```
RequestDate:2008-09-28
ResponseDueDate:2008-10-13
FormNumber:UIB-290e
```

The State Pull Data File Model Connector expects two command line arguments:

Table 46 – Spring State Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Response Collection Query.

To execute the Model Connector from the command line, type:

- *java org.uisides.client.state.StatePullClientDataFile SI|EV Data_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.state.StatePullClientDataFile SI data/StateSIPullQuery.txt*

where:

- SI is the exchange the file is destined for

Example State Pull File

```
#SOAP Header Values
From:ST
To:Broker
PullCollection:3
StateSOAPTransactionNumber:141690


#pull query values
#mandatory field, same as From value
StatePostalCode:ST
#optional fields based on PullCollection value
StateSOAPTransactionNumber:141690
BrokerRecordEffectiveDateFrom:2010-07-13T00:00:00
BrokerRecordEffectiveDateTo:2010-07-14T00:00:00
```

7.2.2.2 Employer/TPA Model Connector – Spring WS

This sample Model Connector demonstrates how an Employer/TPA can access the UI **SIDES** Broker Web services using Spring-WS Model Connector libraries.

REDACTED support is delegated by Spring-WS to Apache Wss4j.

This Model Connector also uses JAXB2 library to marshall/unmarshall XML to/from Java beans.

The Model Connector was developed on JDK 1.5 (Java 5), but should also run on jdk 1.4 and Java 6.

The following main libraries are used:

- Spring-2.5.6 (Core Spring library)
- Spring-ws-1.5.8 (Spring Web Services library)
- Apache Wss4j-1.5.8 (**REDACTED** provider)
- Stax-api-1.0.1 (Streaming API for XML)
- JAXB2 2.1.7 (JAXB2 marshaller/unmarshaller)

For convenience, this sample includes all necessary Eclipse project config files and can be imported into an existing Eclipse IDE workspace. Eclipse 3.5 (Galileo) or later is required.

7.2.2.2.1 Sample Folders and Files

Root folder: sides-employer-client

- ./build.xml

Ant build file (requires Apache Ant 1.7.1 or later)
  - Run "ant build" to compile
  - Run "ant run-post" to execute sample Employer/TPA Post ws call
  - Run "ant run-pull" to execute sample Employer/TPA Pull/Employer/TPA Pull Acknowledgement ws calls

- ./run-post.*

  Unix/Windows shell scripts to run sample Employer/TPA Post Model Connector
  - build sample with "ant build" first

- ./run-pull.*

  Unix/Windows shell scripts to run sample Employer/TPA Pull Model Connector
  - build sample with "ant build" first

- ./src

  Contains:
  - Java source code
  - The Spring config xml file (employer-ws-emulator-config.xml)
  - Log4j config file (log4j.properties)
  - Sample

- ./lib
  Contains required library jar files. All libraries used are open-source Apache LGPL-style libraries which can be freely distributed.

- ./schemas –

  Contains UI **SIDES** XML schema files and Employer/TPA WSDL file

- ./data

  Contains sample payload xml data files for Employer/TPA Post (EmpPost.xml) and Employer/TPA Pull Query (EmployerPullQuery1.xml)

- ./bin

  Build destination folder for compiled Java class files.

7.2.2.2.2 RunTime Configuration

The Model Connector has runtime configuration parameters that allow the employer/TPA to setup its connector. The configuration is specified in the Spring config xml file. The bean that specifies these parameters is the configParams bean. All Java Application Model Connector classes use the same Spring configuration file, ***employer-ws-emulator-config.xml***.

Table 47 - ConfigParam options

Parameter Name	**Applies To**	**Definition**
debugLogFilePath	Post and Pull	The fully qualified location of the debug log file; it contains all the information in all the log files plus detailed information on the state of the Model Connectors workings.
resultsLogFilePath	Post and Pull	The fully qualified location of the results log file; it contains all the information with the results from the call to the Broker.
brptLogFilePath	Post and Pull	The fully qualified location of the brpt log file; it contains all the information with the results from the BRPT.
pdfFilePath	Pull	The fully qualified location of the PDF file and attachments; it will contain all the response received in PDF form with all of the attachments decoded and stored in the same directory
writeRequestsAsPDF	Pull	A boolean value that is “true” if the requests should be printed out as the PDF and “false” otherwise.
requestFlatFilePath	Pull	The fully qualified location of the flat file containing the Request information; it will contain all the requests received in flat file format with all of the attachments still encoded
writeRequestsAsFlatFile	Pull	A boolean value that is “true” if the requests should be written in the flat file format and “false” otherwise.
pullAllFiles	Pull	A boolean value that is “true” if the connector wants the system to pull all files until the message code = 2 and “false” if the connector wants to make the call repeatedly (so as to allow the connector more control).

7.2.2.2.3 Web Services Configuration

The URL of the **SIDES** Broker Web services and **REDACTED** configuration is specified in the Spring config xml file. All Java Application Model Connector classes use the same Spring configuration file, ***employer-ws-emulator-config.xml***.

The sample Model Connector uses a **REDACTED**

1. Import the **REDACTED** file
   - See the article **REDACTED** for helpful hints
2. Update **REDACTED**
3. Send the updated information on the **REDACTED** to the Broker.

7.2.2.2.4 Execution

This Model Connector contains four top-level Java classes with a main() method:

- EmployerPostClient
- EmployerPullClient
- EmployerPostClientDataFile
- EmployerPullClientDataFile

7.2.2.2.4.1 EmployerPostClient/EmployerPullClient

These files implement the Employer/TPA Post and the Employer/TPA Pull/Pull Acknowledgment Web service calls respectively.

The EmployerPostClient and EmployerPullClient use JAXB2 library for manipulating XML elements as Java beans.

The Model Connectors read message payload content from XML files in the designated folder, send it to the test **SIDES** Broker Web services URL, and log Broker's response to the console.

The employer/TPA Post Model Connector expects five command line arguments:

Table 48 – Spring Employer/TPA Post Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification

Model Connector Argument	Definition
"FROM" SOAP header	This is the unique id of the Employer/TPA that is sending the file.
"TO" SOAP header	This is the unique id of the State that the file is destined for.
"EmployerTPAResponseFileGUID" SOAP header	This is the Employer/TPA Response File GUID.
The payload XML source file	The XML file that contains the payload for the call.

To execute the Model Connector from the command line, type:

- *java org.uisides.client.employer.EmployerPostClient SI|EV FROM TO EmployerTPARequestFileGUID Payload_XML_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.employer.EmployerPostClient SI BR999999999 ST 12345678901234567890123456789012 data/ResponseSI1.xml*

where:

- SI is the exchange the file is destined for
- BR999999999 is the Employer Test unique id
- ST is the unique id for State Test UI **SIDES** endpoint
- 12345678901234567890123456789012 is a test EmployerTPAResponseFileGUID.

The employer/TPA Pull Model Connector expects five or six command line arguments:

Table 49 – Spring Employer/TPA Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
"FROM" SOAP header	This is the unique id of the State that is sending the file.
"TO" SOAP header	This must be “Broker.”
“PullCollection” SOAP header	1 for regular pull, 2 for re-pull by transmission number, 3 for re-pull by date range
The payload XML source file	The XML file that contains the payload for the call.
EmployerTPASOAPTransactionNumber	The value of the

Model Connector Argument	Definition
(optional)	"EmployerTPASOAPTransactionNumber" SOAP header. Only required if the pullCollection parameter is 2 (re-pull by transaction number) or 3 (re-pull by date range)

To execute the Model Connector from the command line, type:

- *java org.uisides.client.employer.EmployerPullClient SI|EV FROM* ***Broker*** *PullCollection Payload_XML_File_Name [EmployerTPASOAPTransactionNumber]*

Sample Model Connector arguments are:

- *java org.uisides.client.employer.EmployerPullClient SI BR999999999 Broker 1 data/EmployerSIPullQuery.xml*

where:

- SI is the exchange the file is destined for
- BR999999999 is the Employer Test unique id
- 1 is the pull collection

7.2.2.2.4.2 EmployerPostClientDataFile/EmployerPullClientDataFile

These files implement the employer/TPA Post and the employer/TPA Pull/Pull Acknowledgment Web service calls that read ASCII files. See Figure 1 and Figure 2.

The EmployerPostClientDateFile/EmployerPullClientDataFile Model Connectors read message payload content from ASCII flat files in the designated folder, send it to the test **SIDES** Broker Web services URL, and log Broker's response to the console.

The employer/TPA Post Data File Model Connector expects two command line arguments:

Table 50 – Spring Employer/TPA Post Data File Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Data File	This is the fully qualified path and name of

Model Connector Argument	Definition
	the data file that contains the flat file structure of the Response(s).

To execute the Model Connector from the command line, type:

- *java org.uisides.client.employer.EmployerPostClientDataFile SI|EV Data_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.employer.EmployerPostClientDataFile SI data/EmployerSIPost.txt*

where:

- SI is the exchange the file is destined for

Example Employer Response File

```
#SOAP Headers
To:ST
From:BR999999999
FileGuid:12345678901234567890123456789014

StateRequestRecordGUID:30000000000000000000000000004003
BrokerRecordTransactionNumber:2013889
SSN:560348477
ClaimEffectiveDate:2007-06-04
ClaimNumber:388620
StateEmployerAccountNbr:0065560
CorrectedEmployerName:J C Penny
CorrectedStateEmployerAccountNbr:0123456789
CorrectedFEIN:987654321
OtherSSN:660348477
ClaimantNameWorkedAsForEmployer:Andy Wilson
ClaimantJobTitle:Customer Service Associate
SeasonalEmploymentInd:N
EmployerReportedClaimantFirstDayofWork:2007-10-11
EmployerReportedClaimantLastDayofWork:2007-10-14
EffectiveSeparationDate:2007-10-14
TotalEarnedWagesNeededInd:3
TotalWeeksWorkedNeededInd:3
AverageWeeklyWage:125.00
EmployerSepReasonCode:3
ReturnToWorkInd:N
WorkingAllAvailableHoursInd:N
NotWorkingAvailableHoursReason:NotWorkingAvailableHoursReason
LaborDisputeTypeInd:L

#Remuneration
RemunerationTypeCode:3
RemunerationAmountPerPeriod:999.99
```

```
RemunerationPeriodFrequencyCode:B
DateRemunerationIssued:2007-10-15
EmployerAllocationInd:N
AllocationBeginDate:2007-10-15
AllocationEndDate:2007-10-22

AverageNumberofHoursWorkedperWeek:40
MandatoryRetirementInd:N
MandatoryPension:N
ContributoryorNotContributoryClaimantInd:N
ClaimantPensionContributionPercent:100
DischargeReasonCode:3
FinalIncidentReason:FinalIncidentReason
FinalIncidentDate:2007-10-13
ViolateCompanyPolicyInd:N
DischargePolicyAwareInd:N
DischargePolicyAwareExplanationCode:V

#PriorIncidentOccurrence
PriorIncidentDate:2007-10-10
PriorIncidentReason:None
PriorIncidentWarningInd:Y
PriorIncidentWarningDate:2007-10-10
PriorIncidentWarningDescription:Verbal

WhoDischargedName:Andy Wilson
WhoDischargedTitle:Customer Service Associate
VoluntarySepReasonCode:3
HiringAgreementChangesCode:3
HiringAgreementChangesComments:HiringAgreementChangesComments
ClaimantActionsToAvoidQuitInd:N
ContinuingWorkAvailableInd:N
PreparerTypeCode:T
PreparerCompanyName:J C Penny
PreparerTelephoneNumberPlusExt:9724312108
PreparerContactName:Ed A Jones
PreparerTitle:Project Manager
PreparerFaxNbr:9725312108
PreparerEmailAddress:edjones@jcpenneytest.com
```

The employer/TPA Pull Data File Model Connector expects two command line arguments:

Table 51 – Spring Employer/TPA Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Request Collection Query.

To execute the Model Connector from the command line, type:

- *java org.uisides.client.state.EmployerPullClientDataFile SI|EV Data_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.state.EmployerPullClientDataFile SI data/StateSIPullQuery.txt*

where:

  - SI is the exchange the file is destined for

Example Employer/TPA Pull File

```
#SOAP Header Values
From:BR000000003
To:Broker
PullCollection:3
EmployerTPASOAPTransactionNumber:141690

#pull query values
#mandatory field, same as From value
UniqueID:BR000000003
#optional fields based on PullCollection value
EmployerTPASOAPTransactionNumber:141690
BrokerRecordEffectiveDateFrom:2010-07-13T00:00:00
BrokerRecordEffectiveDateTo:2010-07-14T00:00:00
```

7.2.3 Model Connector - .Net (C#)

7.2.3.1 State Model Connector – .Net (C#)

This Model Connector class library and Windows console application demonstrates how a state and employer connector can access the UI **SIDES** Broker Web services using a Windows .Net client application written in C#.

The Model Connector was originally developed in C# using Visual Studio 2005 for .Net Framework 2.0 with .Net 3.0 runtime components and WCF. The current version of the software was refactored and updated for Visual Studio Express 2010, C# Edition.

In order to open the sample application in Visual Studio, the following system requirements must be met:

- Visual Studio Express 2010 or later, C# Edition.
- Microsoft Windows SDK for .Net Framework 3.0 or later

7.2.3.1.1 Sample Folders and Files

Root folder: sides-state-client-wcf

- ./StateClient.sln
  - Visual Studio Express 2010 solution file. Double click to open Model Connector projects inside Visual Studio.
  - Solution contains two projects: StateClient, a class library project that implements functionality for calling Broker Web services, and StateClientConsole, a Windows Console application project which wraps StateClient class library to provide access to its functionality via command prompt.
- ./StateClient.csproj
  - Visual Studio Express 2010 C# Project file for StateClient class library. Produces stateClient.dll executable class library.
- ./StateClient.cs, ./EvStateClient.cs
  - C# source code file – main source code file for StateClient class library.
- ./util/XmlSerializerUtils.cs
  - C# source code file containing static utility methods for reading/writing XML files using XmlSerializer.
- ./util/DataFileReader.cs, ./util/DataFileWriter.cs
  - C# source code files for the data file reader and writer
- .util/ClientUtils.cs
  - C# source code file for the client utilities package
- ./Service References/StateBroker.cs, ./Service References/EvStateBroker.cs
  - C# source code file containing the WCF Web service client implementation for State Broker Web services. Generated by ServiceModel Metadata Utility Tool (Svcutil.exe) Windows SDK tool from StateBroker.wsdl file and referenced XSD (XML Schema) files.
- ./Service References/StateBroker.map, ./Service References/EvStateBroker.map
  - Visual Studio file generated using “Add Service Reference” tool. Maps Test Broker Web services URL to generated ServiceBroker.cs interfaces.
- ./App.config
  - Main application configuration file, becomes stateClient.dll.config when project is built.
- ./gen_svc_ref.bat
  - Windows shell scripts to generate ./Service References/StateBroker.cs WCF client code for State Broker Web services using Svcutil.exe Windows SDK tool
- ./StateClientConsole
  - Contains console interface wrapper C# project files for StateClientConsole, a Windows console application project which provides command-line interface for the StateClient class library
- ./StateClientConsole/StateClientConsole.csproj
  - StateClientConsole C# Windows console application project file. Part of StateClient.sln solution.
- ./StateClientConsole/StateClientConsole.csproj
  - StateClientConsole source code file. Parses command-line arguments and calls method in stateClient.dll
- ./StateClientConsole/App.config

    - Console interface application configuration file, copy of ./App.config. Becomes stateClientConsole.exe.config in output binaries.
- ./lib
    - Contains log4net.dll class library – Apache Log4Net open-source logging library
- ./data
    - Contains StateBroker.wsdl, XSD XML schema files referenced in the WSDL file, sample Separation Request and Pull query XML files.
- ./bin, ./obj
    - Build and debugging artifact destination folders for compiled application and DLL files.

./

7.2.3.1.2 Run Time Configuration

The Model Connector has runtime configuration parameters that allow the state to setup its connector. These parameters are identified in the AppSettings section of the AppConfig file. In Visual Studio at development time this file is named ./App.config (for StateBroker.dll) and ./StateClientConsole/App.config (for console interface). All versions of this file have identical content. The keys used by the runtime configuration are detailed below.

Table 52 - AppSettings options

Key Name	Applies To	Definition
debugLogFilePath	Post and Pull	The fully qualified location of the debug log file; it contains all the information in all the log files plus detailed information on the state of the Model Connectors workings.
resultsLogFilePath	Post and Pull	The fully qualified location of the results log file; it contains all the information with the results from the call to the Broker.
brptLogFilePath	Post and Pull	The fully qualified location of the brpt log file; it contains all the information with the results from the BRPT.
pinLogFilePath	Post	The fully qualified location of the pin log file; it contains all the information on the new pin created if the *createPin* config parameter is set to true
pdfFilePath	Pull	The fully qualified location of the PDF file and attachments; it will contain all the response received in PDF form with all of the attachments decoded and stored in the same directory
writeResponsesAsPDF	Pull	A boolean value that is “true” if the responses should

Key Name	Applies To	Definition
		be printed out as the PDF and “false” otherwise.
responseFlatFilePath	Pull	The fully qualified location of the flat file containing the Response information; it will contain all the responses received in flat file format with all of the attachments still encoded
writeResponsesAsFlatFile	Pull	A boolean value that is “true” if the responses should be written in the flat file format and “false” otherwise.
createPin	Post	A boolean value that is “true” if the system is directed to create the PIN for the request and “false” otherwise.
pullAllFiles	Pull	A boolean value that is “true” if the connector wants the system to pull all files until the message code = 2 and “false” if the connector wants to make the call repeatedly (so as to allow the connector more control).

7.2.3.1.3 Web Services Configuration

Configuration parameters such as Broker Web services URL and settings are located in the StateClient.dll.config (if only stateClient.dll is used), as well as in StateClientConsole.exe.config if the command-line interface is used. In Visual Studio at development time this file is named ./App.config (for stateBroker.dll) and ./StateClientConsole/App.config (for console interface). All versions of this file have identical content.

Notable configuration settings are:

- address attribute of the endpoint element: determines the Broker Web services URL used by the client. By default, Test Broker Web services URL is used: <endpoint address=” https:// **REDACTED**” …/>.

- httpTransport or httpsTransport child elements of binding element: one of these elements must be present, and must reflect the URL type in endpoint address attribute as described above. Use <httpTransport/> for non-secure Web service URL like http://localhost:8080/sides-trunk/ws, and <httpsTransport authenticationScheme="Negotiate" maxReceivedMessageSize="10000000"/> for secure URLs. httpsTransmport element is enabled by default to correspond to the secure Test Broker URL (https **REDACTED**).

- findValue attribute of **REDACTED**

- findValue attribute **REDACTED**

In order to be able to run this Model Connector, you must first install the client

REDACTED

7.2.3.1.4 Build and Execution

To build Model Connector executables, run “Build->Rebuild Solution” from Visual Studio menu. The StateClientConsole project is configured as the startup project, so if you select Debug->Start Debugging from Visual Studio menu, org.uisides.client.state.StateClientConsole’s main() method will invoked, defined in ./StateClientConsole/StateClientConsole.cs. Since the method expects command-line arguments, the arguments can be specified by right-clicking the StateClientConsole project node inside Solution Explorer, then selecting Properties. In the properties page that opens, select the Debug tab and type the parameters in the “Command line arguments” field.

The command-line interface expects the following types of arguments.

For State Post, the arguments are:

stateClientConsole post SI|EV From To StateRequestFileGUID payloadFileName [SEIN PIN]

Table 53 – .Net (C#) State Post Model Connector Command Line Arguments

Model connector Argument	Definition
Post	The operation name indicating State Post
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
From	The value of the “From” SOAP header, which is the unique ID of the State as defined by the Broker admin, for example “ST” for State Test endpoint
To	The value of the “To” SOAP header, the unique ID of the destination employer/TPA as defined by the Broker admin, for example “BR999999999” for Employer Test endpoint
StateRequestFileGUID	The value of the “StateRequestFileGUID” SOAP header, for example “12345678901234567890123456789012”
payloadFileName	The path to the XML file with StateSeparationRequestCollection as the root element
SEIN (Optional)	The SEIN (State employer identification number) for the destination employer/TPA. Only required if the destination employer/TPA is expected to use the SIDES Employer Website(SEW) (not regular SIDES Web services) to provide a response.
PIN (Optional)	The PIN (personal identification number) for the destination employer/TPA to use when accessing the SEW Website. Only required if the destination employer/TPA is expected to use the SEW Website (not regular SIDES Web services) to provide a response.

Here is an example command for State Test endpoint posting/sending data in data/StatePost.xml file to Employer Test endpoint (BR999999999):

> *stateClientConsole post SI ST BR999999999 12345678901234567890123456789012 data\StatePost.xml*

The State client will send the content of data\StatePost.xml in a properly secured SOAP message with specified header values, and will print out the response payload from Broker (an acknowledgment) along with any response SOAP header values. The response to the example command should look like this:

```
2010-07-07 19:09:28,894 [1] DEBUG [(null)] - Response From header:
Broker
2010-07-07 19:09:28,894 [1] DEBUG [(null)] - Response To header: ST
2010-07-07 19:09:28,894 [1] DEBUG [(null)] - Response
StateRequestFileGUID header: 12345678901234567890123456789012
2010-07-07 19:09:28,894 [1] DEBUG [(null)] - Response MessageCode
header: 1
2010-07-07 19:09:29,067 [1] DEBUG [(null)] - Response: <?xml
version="1.0"?>
<StateSeparationRequestCollectionAcknowledgement
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <StateRequestFileGUID xmlns="https:// REDACTED
/schemas">12345678901234567890123456789012</StateRequestFileGUID>
  <NumberOfRequestRecordsReceived xmlns="https:// REDACTED
/schemas">1</NumberOfRequestRecordsReceived>
  <NumberOfRequestRecordsInError xmlns="https:// REDACTED
/schemas">0</NumberOfRequestRecordsInError>
  <DateStartedReceivingTransmission xmlns="https:// REDACTED
/schemas">2010-07-07T19:09:28.489-
05:00</DateStartedReceivingTransmission>
  <DateFinishedReceivingTransmission xmlns="https:// REDACTED
/schemas">2010-07-07T19:09:28.718-
05:00</DateFinishedReceivingTransmission>
</StateSeparationRequestCollectionAcknowledgement>
2010-07-07 19:09:29,068 [1] DEBUG [(null)] - Press any key to exit.
```

For State Pull, the arguments are:

stateClientConsole pull SI|EV From To pullCollection payloadFileName [stateSoapTnNumber]

Table 54 – .Net (C#) State Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
Pull	The operation name indicating State Pull
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
From	The value of the "From" SOAP header, which is the unique ID of the State as defined by the Broker admin, for example

Model Connector Argument	Definition
	“ST” for State Test endpoint
To	The value of the “To” SOAP header, which is always “Broker” for pull operations
pullCollection	The value of the “PullCollection” SOAP header indicating type of pull, one of: 1 for regular pull, 2 for re-pull by transmission number, 3 for re-pull by date range
payloadFileName	The path to the XML file
stateSoapTnNumber (optional)	The value of the “StateSOAPTransactionNumber” SOAP header. Only required if the pullCollection parameter is 2 (re-pull by transaction number) or 3 (re-pull by date range)

Here is an example command for State Test endpoint pulling any staged responses (regular pull, pullCollection = 1) based on query parameters in data/StatePullQuery.xml:

stateClientConsole pull SI ST Broker 1 data\StatePullQuery.xml

The State client will send the content of data\StatePullQuery.xml in a properly secured SOAP message with specified header values, and will print out the response payload from Broker (a collection of responses if any) along with any response SOAP header values. The client will then print out and send and acknowledgement to Broker to acknowledge pulled responses. The response to the example command should look like this (an empty responses collection was pulled in this case):

```
2010-07-07 19:12:33,477 [1] DEBUG [(null)] - Response From header:
Broker
2010-07-07 19:12:33,477 [1] DEBUG [(null)] - Response To header: ST
2010-07-07 19:12:33,477 [1] DEBUG [(null)] - Response
StateSOAPTransactionNumber header: 48302
2010-07-07 19:12:33,477 [1] DEBUG [(null)] - Response
NextStateSOAPTransactionNumber header:
2010-07-07 19:12:33,477 [1] DEBUG [(null)] - Response MessageCode
header: 2
2010-07-07 19:12:33,696 [1] DEBUG [(null)] - Response: <?xml
version="1.0"?>
<StateSeparationResponseCollection
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
2010-07-07 19:12:34,094 [1] DEBUG [(null)] - Sent acknowledgment: <?xml
version="1.0"?>
<StateSeparationResponseCollectionAcknowledgement
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <StateSOAPTransmissionNumber xmlns="https:// REDACTED
/schemas">48302</StateSOAPTransmissionNumber>
  <NumberOfResponseRecordsReceived xmlns="https:// REDACTED
/schemas">0</NumberOfResponseRecordsReceived>
  <NumberOfResponseRecordsInError xmlns="https:// REDACTED
/schemas">0</NumberOfResponseRecordsInError>
```

```
  <DateStartedReceivingTransmission xmlns="https:// REDACTED
/schemas">0001-01-01T00:00:00</DateStartedReceivingTransmission>
  <DateFinishedReceivingTransmission xmlns="https:// REDACTED
/schemas">0001-01-01T00:00:00</DateFinishedReceivingTransmission>
</StateSeparationResponseCollectionAcknowledgement>
2010-07-07 19:12:34,095 [1] DEBUG [(null)] - Press any key to exit.
```

7.2.3.1.4.1 StateClientConsole with ASCII file

This file implements the State Post and the State Pull/Pull Acknowledgment Web service calls respectively that read ASCII files. See Figure 1 and Figure 2.

The StateClientConsole with ASCII file Model Connectors reads message payload content from ASCII flat files in the designated folder, sends it to the test **SIDES** Broker Web services URL, and logs the Broker's response to the console/log files.

The State Post Model Connector expects three command line arguments:

Table 55 – .Net State Post Data File Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Post	This is a Post command
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Separation Request(s).

To execute the Model Connector from the command line, type:

- *StateClientConsole SI|EV post Data_File_Name*

Sample Model Connector arguments are:

- *StateClientConsoleDataFile SI post data/StateSIPost.txt*

where:

- SI is the exchange the file is destined for

Example State Request File

```
#SOAP Header Values
To:BR999999999
```

```
From:ST
FileGuid:01234567890123456789012345678901

StateRequestRecordGUID:c755d30ed7dd4662bc0452e9050c00df
SSN:000989494
ClaimEffectiveDate:2008-09-28
ClaimNumber:0
StateEmployerAccountNbr:342424001
EmployerName:ELDORA ENTERPRISES LTD LIABILITY CO
FEIN:841173055
TypeofEmployerCode:1
TypeofClaimCode:1
BenefitYearBeginDate:2008-09-28
RequestingStateAbbreviation:CO
UIOfficeName:CO CDLE
UIOfficePhone:3033189055
UIOfficeFax:3033189014
ClaimantLastName:WHEELOCK
ClaimantFirstName:PHILIPPE
ClaimantMiddleInitial:M
ClaimantJobTitle:SKI PATROL
ClaimantReportedFirstDayofWork:2005-11-25
ClaimantReportedLastDayofWork:2008-04-10
WagesWeeksNeededCode:NA
ClaimantSepReasonCode:1

UniqueAttachmentId:01
DescriptionofAttachmentCode:1
TypeofDocument:test-file.txt
ActionableAttachment:3
AttachmentSize:2000
```

```
AttachmentData:QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQU
FBDQpCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCDQp
DQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDDQpERERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERQNCkVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUUNCkZGRkZGRkZGRkZGRkZGRk
ZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkYNCkdHR0dHR0dHR0dHR0dHR0dHR0d
HR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0cNCkhISEhISEhISEhISEhISEhISEhISEhISEhISEhI
SEhISEhISEhISEhISEhISEhISEhISEgNCklJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJS
UlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJDQpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSk
pKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSg0KS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0t
LS0tLS0tLS0tLS0tLS0tLSw0KTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExM
TExMTExMTExMTEwNCk1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU0NCk5OT
k5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk4NCk9PT09PT09PT0
9PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT08NClBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQDQpRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRDQpSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSU
lJSUlJSUlJSUlJSUlJSUlJSUlJSUg0KU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1
NTU1NTU1NTU1NTU1NTU1MNClRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUDQpBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUENCkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkINC
kNDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0MNCkRERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERA0KRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRQ0KRkZGRkZGRkZGRkZGRkZG
RkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRg0KR0dHR0dHR0dHR0dHR0dHR0dHR
0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHRw0KSEhISEhISEhISEhISEhISEhISEhISEhISEhISE
```

```
hISEhISEhISEhISEhISEhISEhISEhISA0KSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUl
JSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUkNCkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpK
SkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKDQpLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS
0tLS0tLS0tLS0tLS0tLS0tLDQpMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE
xMTExMTExMTExMTA0KTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTQ0KTk5
OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTg0KT09PT09PT09P
T09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PTw0KUFBQUFBQUFBQUFBQUFBQU
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFANClFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVENClJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJ
SUlJSUlJSUlJSUlJSUlJSUlJSUlJSDQpTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NT
U1NTU1NTU1NTU1NTU1NTUw0KVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFQNCg==
```

UniqueAttachmentId:02
DescriptionofAttachmentCode:1
TypeofDocument:test-file2.txt
ActionableAttachment:3
AttachmentSize:2000

```
AttachmentData:QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQU
FBDQpCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCDQp
DQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDDQpERERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERQNCkVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUUNCkZGRkZGRkZGRkZGRkZGRk
ZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkYNCkdHR0dHR0dHR0dHR0dHR0dHR0d
HR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0cNCkhISEhISEhISEhISEhISEhISEhISEhISEhI
SEhISEhISEhISEhISEhISEhISEgNCklJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJS
UlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJDQpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSk
pKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSg0KS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0t
LS0tLS0tLS0tLS0tLS0tLSw0KTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExM
TExMTExMTExMTEwNCk1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU0NCk5OT
k5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk4NCk9PT09PT09PT0
9PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT08NClBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQDQpRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRDQpSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSU
lJSUlJSUlJSUlJSUlJSUlJSUlJSUg0KU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1
NTU1NTU1NTU1NTU1NTU1MNClRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUDQpBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUENCkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkINC
kNDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0MNCkRERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERA0KRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRQ0KRkZGRkZGRkZGRkZG
RkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRg0KR0dHR0dHR0dHR0dHR0dHR0dHR0dHR
0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHRw0KSEhISEhISEhISEhISEhISEhISEhISEhISEhISE
hISEhISEhISEhISEhISEhISEhISEhISEhISA0KSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUl
JSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUkNCkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpK
SkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKDQpLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS
0tLS0tLS0tLS0tLS0tLS0tLDQpMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE
xMTExMTExMTExMTA0KTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTQ0KTk5
OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTg0KT09PT09PT09P
T09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PTw0KUFBQUFBQUFBQUFBQUFBQU
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFANClFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVENClJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJ
SUlJSUlJSUlJSUlJSUlJSUlJSUlJSDQpTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NT
U1NTU1NTU1NTU1NTU1NTUw0KVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFQNCg==
RequestDate:2008-09-28
ResponseDueDate:2008-10-13
FormNumber:UIB-290e
```

The State Pull Client Model Connector expects three command line arguments:

Table 56 – .Net State Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Pull	This is a Pull command
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Response Collection Query.

To execute the Model Connector from the command line, type:

- *StateClientConsole SI|EV pull Data_File_Name*

Sample Model Connector arguments are:

- *StateClientConsole SI pull data/StateSIPullQuery.txt*

where:

- SI is the exchange the file is destined for

Example State Pull File

```
#SOAP Header Values
From:ST
To:Broker
PullCollection:3
#StateSOAPTransactionNumber:141690


#pull query values
#mandatory field, same as From value
StatePostalCode:ST
#optional fields based on PullCollection value
#StateSOAPTransactionNumber:141690
BrokerRecordEffectiveDateFrom:2010-07-13T00:00:00
```

```
BrokerRecordEffectiveDateTo:2010-07-14T00:00:00
```

7.2.3.2 Employer Model Connector – .Net (C#)

This Model Connector class library and Windows console application demonstrates how an employer/TPA connector can access the UI **SIDES** Broker Web services using a Windows .Net client application written in C#.

The Model Connector was originally developed in C# using Visual Studio 2005 for .Net Framework 2.0 with .Net 3.0 runtime components and WCF. The current version of the software was refactored and updated for Visual Studio Express 2010, C# Edition.

In order to open the sample application in Visual Studio, the following system requirements must be met:

- Visual Studio Express 2010 or later, C# Edition.
- Microsoft Windows SDK for .Net Framework 3.0 or later

7.2.3.2.1 Sample Folders and Files

Root folder: sides-employer-client-wcf

- ./EmployerClient.sln
  - Visual Studio Express 2010 solution file. Double click to open Model Connector projects inside Visual Studio.
  - Solution contains two projects: EmployerClient, a class library project that implements functionality for calling Broker Web services, and EmployerClientConsole, a Windows Console application project which wraps EmployerClient class library to provide access to its functionality via command prompt.
- ./EmployerClient.csproj
  - Visual Studio Express 2010 C# Project file for EmployerClient class library. Produces EmployerClient.dll executable class library.
- ./EmployerClient.cs, ./EvEmployerClient.cs
  - C# source code file – main source code file for EmployerClient class library.
- ./util/XmlSerializerUtils.cs
  - C# source code file containing static utility methods for reading/writing XML files using XmlSerializer.
- ./util/DataFileReader.cs, ./util/DataFileWriter.cs
  - C# source code files for the data file reader and writer
- .util/ClientUtils.cs
  - C# source code file for the client utilities package
- ./Service References/EmployerBroker.cs, ./Service References/EvEmployerBroker.cs
  - C# source code file containing the WCF Web service client implementation for Employer Broker Web services. Generated by ServiceModel Metadata Utility

Tool (Svcutil.exe) Windows SDK tool from EmployerBroker.wsdl file and referenced XSD (XML Schema) files.

- ./Service References/EmployerBroker.map, ./Service References/EvEmployerBroker.map
  - Visual Studio file generated using “Add Service Reference” tool. Maps Test Broker Web services URL to generated EmployerBroker.cs interfaces.
- ./App.config
  - Main application configuration file, becomes EmployerClient.dll.config when project is built.
- ./gen_svc_ref.bat
  - Windows shell scripts to generate ./Service References/EmployerBroker.cs WCF client code for Employer Broker Web services using Svcutil.exe Windows SDK tool
- ./EmployerClientConsole
  - Contains console interface wrapper C# project files for EmployerClientConsole, a Windows console application project which provides command-line interface for the EmployerClient class library
- ./EmployerClientConsole/EmployerClientConsole.csproj
  - EmployerClientConsole C# Windows console application project file. Part of EmployerClient.sln solution.
- ./EmployerClientConsole/EmployerClientConsole.csproj
  - EmployerClientConsole source code file. Parses command-line arguments and calls method in EmployerClient.dll
- ./EmployerClientConsole/App.config
  - Console interface application configuration file, copy of ./App.config. Becomes EmployerClientConsole.exe.config in output binaries.
- ./lib
  - Contains log4net.dll class library – Apache Log4Net open-source logging library
- ./data
  - Contains EmployerBroker.wsdl, XSD XML schema files referenced in the WSDL file, sample Separation Responses and Pull query XML files.
- ./bin, ./obj
  - Build and debugging artifact destination folders for compiled application and DLL files.
- ./ **REDACTED**

7.2.3.2.2 Run Time Configuration

The Model Connector has runtime configuration parameters that allow the employer/TPA to setup its connector. These parameters are identified in the AppSettings section of the AppConfig file. In Visual Studio, at development time, this file is named ./App.config (for EmployerBroker.dll) and ./EmployerClientConsole/App.config (for console interface). All versions of this file have identical content. The keys used by the runtime configuration are detailed below.

Table 57 - AppSettings options

Key Name	Applies To	Definition
debugLogFilePath	Post and Pull	The fully qualified location of the debug log file; it contains all the information in all the log files plus detailed information on the state of the Model Connectors workings.
resultsLogFilePath	Post and Pull	The fully qualified location of the results log file; it contains all the information with the results from the call to the Broker.
brptLogFilePath	Post and Pull	The fully qualified location of the brpt log file; it contains all the information with the results from the BRPT.
pdfFilePath	Pull	The fully qualified location of the PDF file and attachments; it will contain all the requests received in PDF form with all of the attachments decoded and stored in the same directory
writeRequestsAsPDF	Pull	A boolean value that is "true" if the requests should be printed out as the PDF and "false" otherwise.
requestFlatFilePath	Pull	The fully qualified location of the flat file containing the Request information; it will contain all the requests received in flat file format with all of the attachments still encoded
writeRequestsAsFlatFile	Pull	A boolean value that is "true" if the requests should be written in the flat file format and "false" otherwise.
pullAllFiles	Pull	A boolean value that is "true" if the connector wants the system to pull all files until the message code = 2 and "false" if the connector wants to make the call repeatedly (so as to allow the connector more control).

7.2.3.2.3 Web Services Configuration

Configuration parameters such as Broker Web services URL and **REDACTED** settings are located in the EmployerClient.dll.config (if only EmployerClient.dll is used), as well as in EmployerClientConsole.exe.config if the command-line interface is used. In Visual Studio at development time this file is named ./App.config (for EmployerBroker.dll) and ./EmployerClientConsole/App.config (for console interface). All versions of this file have identical content.

Notable configuration settings are:

- address attribute of the endpoint element: determines the Broker Web services URL used by the client. By default, Test Broker Web services URL is used: <endpoint address=" https:// **REDACTED**" …/>

- httpTransport or httpsTransport child elements of binding element: one of these elements must be present, and must reflect the URL type in endpoint address attribute as described above. Use <httpTransport/> for non-secure Web service URL like http://localhost:8080/sides-trunk/ws, and <httpsTransport authenticationScheme="Negotiate" maxReceivedMessageSize="10000000"/> for secure URLs. httpsTransmport element is enabled by default to correspond to the secure Test Broker URL (https:// **REDACTED**)

- findValue **REDACTED**

- findValue **REDACTED**

In order to be able to run this Model Connector, you must first install **REDACTED**

7.2.3.2.4 Build and Execution

To build Model Connector executables, run “Build->Rebuild Solution” from Visual Studio menu. The EmployerClientConsole project is configured as the startup project, so if you select Debug->Start Debugging from Visual Studio menu, org.uisides.client.employer.EmployerClientConsole’s main() method will be invoked, defined in ./StateClientConsole/StateClientConsole.cs. Since the method expects command-line arguments, the arguments can be specified by right-clicking the StateClientConsole project node inside Solution Explorer, then selecting Properties. In the properties page that opens, select the Debug tab and type the parameters in the “Command line arguments” field.

The command-line interface expects the following types of arguments.

For Employer Post, the arguments are:

EmployerClientConsole post SI|EV From To EmployerTPARequestFileGUID payloadFileName

Table 58 – .Net (C#) Employer Post Model Connector Command Line Arguments

Model connector Argument	Definition
Post	The operation name indicating Employer Post
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
From	The value of the “From” SOAP header, which is the unique ID of the Employer as defined by the Broker admin, for

Model connector Argument	Definition
	example “BR999999999” for Employer Test endpoint
To	The value of the “To” SOAP header, the unique ID of the destination State as defined by the Broker admin, for example “ST” for State Test endpoint
EmployerTPAResponseFileGUID	The value of the “EmployerTPAResponseFileGUID” SOAP header, for example “12345678901234567890123456789012”
payloadFileName	The path to the XML file with EmployerTPASeparationResponseCollection as the root element

Here is an example command for Employer Test endpoint posting/sending data in data/EmpPost.xml file to State Test endpoint (ST):

> *EmployerClientConsole post SI BR999999999 ST 12345678901234567890123456789012 data\EmpPost.xml*

The Employer client will send the content of data\EmpPost.xml in a properly secured SOAP message with specified header values, and will print out the response payload from Broker (an acknowledgment) along with any response SOAP header values. The response to the example command should look like this (in this case Broker rejected the only response since it couldn’t find a matching request):

```
2010-07-07 19:37:11,718 [1] DEBUG [(null)] - Response From header: 
Broker
2010-07-07 19:37:11,718 [1] DEBUG [(null)] - Response To header: 
BR999999999
2010-07-07 19:37:11,718 [1] DEBUG [(null)] - Response 
EmployerTPAResponseFileGUID header: 123456778901234567890123456789012
2010-07-07 19:37:11,718 [1] DEBUG [(null)] - Response MessageCode 
header: 2
2010-07-07 19:37:11,875 [1] DEBUG [(null)] - Response: <?xml 
version="1.0"?>
<EmployerTPASeparationResponseCollectionAcknowledgement 
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EmployerTPAResponseFileGUID xmlns="https:// REDACTED
/schemas">123456778901234567890123456789012</EmployerTPAResponseFileGUI
D>
  <FailedSeparationResponse xmlns="https:// REDACTED /schemas">

<StateRequestRecordGUID>c755d30ed7dd4662bc0452e9050c00cd</StateRequestR
ecordGUID>

<BrokerRecordTransactionNumber>27550</BrokerRecordTransactionNumber>
    <ErrorOccurrence>
      <ErrorCode>210</ErrorCode>
      <ErrorMessage>Business Rule violation - There is no matching 
Claim Request record with fields matching A1 to B1, A2 to B2, A3 to B3,
A4 to B4, the StateRequestRecordGUID, and the
BrokerRecordTransactionNumber.</ErrorMessage>
    </ErrorOccurrence>
  </FailedSeparationResponse>
  <NumberOfResponseRecordsReceived xmlns="https:// REDACTED
/schemas">1</NumberOfResponseRecordsReceived>
  <NumberOfResponseRecordsInError xmlns="https:// REDACTED
/schemas">1</NumberOfResponseRecordsInError>
  <DateStartedReceivingTransmission xmlns="https:// REDACTED
/schemas">2010-07-07T19:37:09.343-
05:00</DateStartedReceivingTransmission>
  <DateFinishedReceivingTransmission xmlns="https:// REDACTED
/schemas">2010-07-07T19:37:11.557-
05:00</DateFinishedReceivingTransmission>
</EmployerTPASeparationResponseCollectionAcknowledgement>
2010-07-07 19:37:11,876 [1] DEBUG [(null)] - Press any key to exit.
```

For Employer Pull, the arguments are:

EmployerClientConsole pull SI|EV From To pullCollection payloadFileName [employerSoapTnNumber]

Table 59 – .Net (C#) Employer Pull Model Connector Command Line Arguments

Model connector Argument	Definition
Pull	The operation name indicating Employer Pull
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
From	The value of the “From” SOAP header, which is the unique ID of the Employer as defined by the Broker admin, for example “BR999999999” for Employer Test endpoint
To	The value of the “To” SOAP header, which is always “Broker” for pull operations
pullCollection	The value of the “PullCollection” SOAP header indicating type of pull, one of: 1, 2 or 3
payloadFileName	The path to the XML file with EmployerTPASeparationRequestCollectionQuery as the root element
employerSoapTnNumber (optional)	The value of the “EmployerTPASOAPTransactionNumber” SOAP header. Only required if the pullCollection parameter is 2 (re-pull by transaction number) or 3 (re-pull by date range)

Here is an example command for Employer Test endpoint pulling any staged responses (regular pull, pullCollection = 1) based on query parameters in data/EmployerPullQuery.xml:

EmployerClientConsole pull SI ST Broker 1 data\EmployerPullQuery.xml

The Employer client will send the content of data\EmployerPullQuery.xml in a properly secured SOAP message with specified header values, and will print out the response payload from Broker (a collection of requests if any) along with any response SOAP header values. The client will then print out and send and acknowledgement to Broker to acknowledge pulled responses. The response to the example command should look like this (an empty requests collection was pulled in this case):

```
2010-07-07 19:45:14,001 [1] DEBUG [(null)] - Response From header: 
Broker
2010-07-07 19:45:14,001 [1] DEBUG [(null)] - Response To header: 
BR999999999
2010-07-07 19:45:14,001 [1] DEBUG [(null)] - Response 
EmployerTPASOAPTransactionNumber header: 48306
2010-07-07 19:45:14,001 [1] DEBUG [(null)] - Response 
NextEmployerTPASOAPTransactionNumber header: 
2010-07-07 19:45:14,002 [1] DEBUG [(null)] - Response MessageCode 
header: 2
2010-07-07 19:45:14,203 [1] DEBUG [(null)] - Response: <?xml 
version="1.0"?>
<EmployerTPASeparationRequestCollection 
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
2010-07-07 19:45:14,448 [1] DEBUG [(null)] - Sent acknowledgment: <?xml 
version="1.0"?>
<EmployerTPASeparationRequestCollectionAcknowledgement 
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EmployerTPASOAPTransmissionNumber xmlns="https:// REDACTED
/schemas">48306</EmployerTPASOAPTransmissionNumber>
  <NumberOfRequestRecordsReceived xmlns="https:// REDACTED
/schemas">0</NumberOfRequestRecordsReceived>
  <NumberOfRequestRecordsInError xmlns="https:// REDACTED
/schemas">0</NumberOfRequestRecordsInError>
  <DateStartedReceivingTransmission xmlns="https:// REDACTED
/schemas">0001-01-01T00:00:00</DateStartedReceivingTransmission>
  <DateFinishedReceivingTransmission xmlns="https:// REDACTED
/schemas">0001-01-01T00:00:00</DateFinishedReceivingTransmission>
</EmployerTPASeparationRequestCollectionAcknowledgement>
2010-07-07 19:45:14,452 [1] DEBUG [(null)] - Press any key to exit.
```

7.2.3.2.4.1 EmployerClientConsole with ASCII file

This file implements the employer/TPA Post and the employer/TPA Pull/Pull Acknowledgment Web service calls that read ASCII files. See Figure 1 and Figure 2.

The EmployerClientConsole Model Connector reads message payload content from ASCII flat files in the designated folder, sends it to the test **SIDES** Broker Web services URL, and logs the Broker's response to the console/log files.

The Employer/TPA Post with ASCII File Model Connector expects three command line arguments:

Table 60 – .Net Employer/TPA Post Data File Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Post	The Post command.
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Separation Response(s).

To execute the Model Connector from the command line, type:

- *EmployerClientConsole SI|EV post Data_File_Name*

Sample Model Connector arguments are:

- *EmployerClientConsole SI post data/EmployerSIPost.txt*

where:

  - SI is the exchange the file is destined for

Example Employer Response File

```
#SOAP Headers
To:ST
From:BR999999999
FileGuid:12345678901234567890123456789014

StateRequestRecordGUID:30000000000000000000000000004003
BrokerRecordTransactionNumber:2013889
SSN:560348477
ClaimEffectiveDate:2007-06-04
ClaimNumber:388620
StateEmployerAccountNbr:0065560
CorrectedEmployerName:J C Penny
CorrectedStateEmployerAccountNbr:0123456789
CorrectedFEIN:987654321
OtherSSN:660348477
ClaimantNameWorkedAsForEmployer:Andy Wilson
ClaimantJobTitle:Customer Service Associate
SeasonalEmploymentInd:N
EmployerReportedClaimantFirstDayofWork:2007-10-11
EmployerReportedClaimantLastDayofWork:2007-10-14
```

```
EffectiveSeparationDate:2007-10-14
TotalEarnedWagesNeededInd:3
TotalWeeksWorkedNeededInd:3
AverageWeeklyWage:125.00
EmployerSepReasonCode:3
ReturnToWorkInd:N
WorkingAllAvailableHoursInd:N
NotWorkingAvailableHoursReason:NotWorkingAvailableHoursReason
LaborDisputeTypeInd:L

#Remuneration
RemunerationTypeCode:3
RemunerationAmountPerPeriod:999.99
RemunerationPeriodFrequencyCode:B
DateRemunerationIssued:2007-10-15
EmployerAllocationInd:N
AllocationBeginDate:2007-10-15
AllocationEndDate:2007-10-22

AverageNumberofHoursWorkedperWeek:40
MandatoryRetirementInd:N
MandatoryPension:N
ContributoryorNotContributoryClaimantInd:N
ClaimantPensionContributionPercent:100
DischargeReasonCode:3
FinalIncidentReason:FinalIncidentReason
FinalIncidentDate:2007-10-13
ViolateCompanyPolicyInd:N
DischargePolicyAwareInd:N
DischargePolicyAwareExplanationCode:V

#PriorIncidentOccurrence
PriorIncidentDate:2007-10-10
PriorIncidentReason:None
PriorIncidentWarningInd:Y
PriorIncidentWarningDate:2007-10-10
PriorIncidentWarningDescription:Verbal

WhoDischargedName:Andy Wilson
WhoDischargedTitle:Customer Service Associate
VoluntarySepReasonCode:3
HiringAgreementChangesCode:3
HiringAgreementChangesComments:HiringAgreementChangesComments
ClaimantActionsToAvoidQuitInd:N
ContinuingWorkAvailableInd:N
PreparerTypeCode:T
PreparerCompanyName:J C Penny
PreparerTelephoneNumberPlusExt:9724312108
PreparerContactName:Ed A Jones
PreparerTitle:Project Manager
PreparerFaxNbr:9725312108
PreparerEmailAddress:edjones@jcpenneytest.com
```

The Employer/TPA Pull Data File Model Connector expects three command line arguments:

Table 61 – .Net Employer/TPA Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Pull	The Pull command
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Request Collection Query.

To execute the Model Connector from the command line, type:

- *EmployerClientConsole SI|EV pull Data_File_Name*

Sample Model Connector arguments are:

- *EmployerClientConsole SI pull data/StateSIPullQuery.txt*

where:

- SI is the exchange the file is destined for

Example Employer/TPA Pull File

```
#SOAP Header Values
From:BR000000003
To:Broker
PullCollection:3
EmployerTPASOAPTransactionNumber:141690

#pull query values
#mandatory field, same as From value
UniqueID:BR000000003
#optional fields based on PullCollection value
EmployerTPASOAPTransactionNumber:141690
BrokerRecordEffectiveDateFrom:2010-07-13T00:00:00
BrokerRecordEffectiveDateTo:2010-07-14T00:00:00
```

7.2.4 Model Connector – JAX-WS

7.2.4.1 State Model Connector – JAX-WS

This sample Model Connector demonstrates how a State can access the UI **SIDES** Broker Web services using JAX-WS libraries.

The Model Connector was developed on JDK 1.5 (Java 5), but should also run on jdk 1.4 and Java 6.

The following main libraries are used (parts of Sun Microsystems Metro v1.4 release):

- JAX-WS RI 2.1.4.1
- JAXB RI 2.1.7.1

For convenience, this sample includes all necessary Eclipse project config files and can be imported into an existing Eclipse IDE workspace. Eclipse 3.5 (Galileo) or later is required.

7.2.4.1.1 Sample Folders and Files

Root folder: sides-state-client-jax-ws

- ./build.xml

  Ant build file (requires Apache Ant 1.7.1 or later)
  - Run “ant generate-jaxws-client” to generate JAX-WS client Java beans from WSDL (creates src/org/uisides/client/state/generated classes)
  - Run "ant build" to compile

- ./src

  Contains:
  - Java source code
  - Log4j config file (log4j.properties)
  - State-ws-client-config.xml – run time settings
  - client-security-env.properties (JAX-WS)

- ./lib

  Contains required library jar files. All libraries used are open-source Apache LGPL-style libraries which can be freely distributed.

- ./schemas –

  Contains UI **SIDES** XML schema files and State WSDL file

- ./data

  Contains sample payload xml data files for State Post (StatePost.xml)

- ./bin

  Build destination folder for compiled Java class files.

7.2.4.1.2 RunTime Configuration

The Model Connector has runtime configuration parameters that allow the state to setup its connector. The configuration is specified in a config xml file. The bean that specifies these parameters is the configParams bean. All Java JAX-WS Application Model Connector classes use the same configuration file, ***state-ws-emulator-config.xml***.

Table 62 - ConfigParam options

Parameter Name	**Applies To**	**Definition**
debugLogFilePath	Post and Pull	The fully qualified location of the debug log file; it contains all the information in all the log files plus detailed information on the state of the Model Connectors workings.
resultsLogFilePath	Post and Pull	The fully qualified location of the results log file; it contains all the information with the results from the call to the Broker.
brptLogFilePath	Post and Pull	The fully qualified location of the brpt log file; it contains all the information with the results from the BRPT.
pinLogFilePath	Post	The fully qualified location of the pin log file; it contains all the information on the new pin created if the *createPin* config parameter is set to true
pdfFilePath	Pull	The fully qualified location of the PDF file and attachments; it will contain all the response received in PDF form with all of the attachments decoded and stored in the same directory
writeResponsesAsPDF	Pull	A boolean value that is “true” if the responses should be printed out as the PDF and “false” otherwise.
responseFlatFilePath	Pull	The fully qualified location of the flat file containing the Response information; it will contain all the responses received in flat file format with all of the attachments still encoded
writeResponsesAsFlatFile	Pull	A boolean value that is “true” if the responses should be written in the flat file format and “false” otherwise.
createPin	Post	A boolean value that is “true” if the system is directed to create the PIN for the request and “false” otherwise.
pullAllFiles	Pull	A boolean value that is “true” if the connector wants the system to pull all files until the message code = 2 and “false” if the connector wants to make the call repeatedly (so as to allow the connector more control).

7.2.4.1.3 Web Services Configuration

The URL of the **SIDES** Broker Web services and **REDACTED** configuration is specified in the client-security-env.properties file.

7.2.4.1.4 Execution

This Model Connector contains four top-level Java class with a main() method:

- StatePostClient
- StatePostClientDataFile
- StatePullClient
- StatePullClientDataFile

These files implement the State Post and State Pull respectively using JAX-WS.

7.2.4.1.4.1 StatePostClient/StatePullClient

The StatePostClient reads Separation Request SOAP message payload content from XML file in the *./data* folder, sends it to the test **SIDES** Broker Web services URL, and logs Broker's responses to the console.

The Model Connector expects five or seven command line arguments:

Table 63 – JAX-WS State Post Model Connector Command Line Arguments

Model connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
"FROM" SOAP header	This is the unique id of the State that is sending the file.
"TO" SOAP header	This is the unique id of the Employer/TPA that the file is destined for.
"StateRequestFileGUID" SOAP header	This is the State Request File GUID.
The payload XML source file	The XML file that contains the payload for the call.
SEIN SOAP header (optional)	SEIN value if sending to **SEW** employer/TPA
PIN SOAP header (optional)	PIN value if sending to **SEW** employer/TPA

To execute the State Post Model Connector from the command line, type:

- *java –cp <classpath> org.uisides.client.state.StatePostClient SI|EV FROM TO StateRequestFileGUID Payload_XML_File_Name [SEIN PIN]*

Sample Model Connector arguments are:

- *java –cp <classpath> org.uisides.client.state.StatePostClient SI ST BR999999999 12345678901234567890123456789012 data/StatePost.xml* where:

- SI is the exchange to file is destined for
- ST is the State Test unique id
- BR999999999 is the unique id for Employer Test UI **SIDES** endpoint
- 12345678901234567890123456789012 is a test StateRequestFileGUID.

The StatePullClient reads State pull query SOAP message payload content from XML file in the *./data* folder, sends it to the test **SIDES** Broker Web services URL, logs Broker's responses to the console, then prepares and sends Broker the acknowledgement of received pull responses.

The Model Connector expects five or six command line arguments:

Table 64 – JAX-WS State Pull Model Connector Command Line Arguments

Model connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
"FROM" SOAP header	This is the unique id of the State that is sending the pull query.
"TO" SOAP header	Always “Broker” for Pull requests
“PullCollection” SOAP header	1 for regular pull, 2 for re-pull by transmission number, 3 for re-pull by date range
The payload XML source file	The XML file that contains the payload for the call.
StateSOAPTransactionNumber (optional)	The value of the “StateSOAPTransactionNumber” SOAP header. Only required if the pullCollection parameter is 2 (re-pull by transaction number) or 3 (re-pull by date range)

To execute the State Pull Model Connector from the command line, type:

- *java –cp <classpath> org.uisides.client.state.StatePullClient SI|EV FROM TO PullCollection Payload_XML_File_Name [StateSOAPTransactionNumber]*

Sample Model Connector arguments are:

- *java –cp <classpath> org.uisides.client.state.StatePostClient SI ST Broker 1 data/StatePullQuery.xml*

where:

- SI is the exchange that the file is destined for

- ST is the State Test unique id
- Broker is the “To” value for all Pull requests
- 1 is the PullCollection value indicating a regular pull

7.2.4.1.4.2 StatePostClientDataFile/StatePullClientDataFile

These files implement the State Post and the State Pull/Pull Acknowledgment Web service calls respectively that read ASCII files. See Figure 1 and Figure 2.

The StatePostClientDateFile/StatePullClientDataFile Model Connectors read message payload content from ASCII flat files in the designated folder, send it to the test **SIDES** Broker Web services URL, and log Broker's response to the console.

The State Post Data File Model Connector expects two command line arguments:

Table 65 – JAX-WS State Post Data File Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Separation Request(s).

To execute the Model Connector from the command line, type:

- *java org.uisides.client.state.StatePostClientDataFile SI|EV Data_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.state.StatePostClientDataFile SI data/StateSIPost.txt*

where:

- SI is the exchange the file is destined for

Example State Request File

```
#SOAP Header Values
To:BR999999999
From:ST
FileGuid:01234567890123456789012345678901

StateRequestRecordGUID:c755d30ed7dd4662bc0452e9050c00df
SSN:000989494
```

```
ClaimEffectiveDate:2008-09-28
ClaimNumber:0
StateEmployerAccountNbr:342424001
EmployerName:ELDORA ENTERPRISES LTD LIABILITY CO
FEIN:841173055
TypeofEmployerCode:1
TypeofClaimCode:1
BenefitYearBeginDate:2008-09-28
RequestingStateAbbreviation:CO
UIOfficeName:CO CDLE
UIOfficePhone:3033189055
UIOfficeFax:3033189014
ClaimantLastName:WHEELOCK
ClaimantFirstName:PHILIPPE
ClaimantMiddleInitial:M
ClaimantJobTitle:SKI PATROL
ClaimantReportedFirstDayofWork:2005-11-25
ClaimantReportedLastDayofWork:2008-04-10
WagesWeeksNeededCode:NA
ClaimantSepReasonCode:1

UniqueAttachmentId:01
DescriptionofAttachmentCode:1
TypeofDocument:test-file.txt
ActionableAttachment:3
AttachmentSize:2000
AttachmentData:QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQU
FBDQpCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCDQp
DQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDDQpERERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERQNCkVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUUNCkZGRkZGRkZGRkZGRkZGRk
ZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkYNCkdHR0dHR0dHR0dHR0dHR0dHR0d
HR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0cNCkhISEhISEhISEhISEhISEhISEhISEhISEhISEhI
SEhISEhISEhISEhISEhISEhISEgNCklJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJS
UlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJDQpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSk
pKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSg0KS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0t
LS0tLS0tLS0tLS0tLS0tLSw0KTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExM
TExMTExMTExMTEwNCk1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU0NCk5OT
k5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk4NCk9PT09PT09PT0
9PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT08NClBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQDQpRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRDQpSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSU
lJSUlJSUlJSUlJSUlJSUlJSUlJSUg0KU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1
NTU1NTU1NTU1NTU1NTU1MNClRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUDQpBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUENCkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkINC
kNDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0MNCkRERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERA0KRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRQ0KRkZGRkZGRkZGRkZG
RkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRg0KR0dHR0dHR0dHR0dHR0dHR0dHR
0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHRw0KSEhISEhISEhISEhISEhISEhISEhISEhISE
hISEhISEhISEhISEhISEhISEhISEhISA0KSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUl
JSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUkNCkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpK
SkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKDQpLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS
0tLS0tLS0tLS0tLS0tLS0tLDQpMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE
xMTExMTExMTExMTA0KTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTQ0KTk5
```

```
OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTg0KT09PT09PT09P
T09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PTw0KUFBQUFBQUFBQUFBQUFBQU
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFANClFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVENClJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJ
SUlJSUlJSUlJSUlJSUlJSUlJSUlJSDQpTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NT
U1NTU1NTU1NTU1NTU1NTUw0KVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFQNCg==

UniqueAttachmentId:02
DescriptionofAttachmentCode:1
TypeofDocument:test-file2.txt
ActionableAttachment:3
AttachmentSize:2000
AttachmentData:QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQU
FBDQpCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCDQp
DQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDDQpERERE
REREREREREREREREREREREREREREREREREREREREREREREREREREREREREREREQNCkVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUUNCkZGRkZGRkZGRkZGRk
ZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkYNCkdHR0dHR0dHR0dHR0dHR0d
HR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0cNCkhISEhISEhISEhISEhISEhISEhISEhISEhI
SEhISEhISEhISEhISEhISEhISEhISEgNCklJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJS
UlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJDQpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSk
pKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSg0KS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0t
LS0tLS0tLS0tLS0tLS0tLSw0KTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExM
TExMTExMTExMTEwNCk1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU0NCk5OT
k5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk4NCk9PT09PT0
9PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT08NClBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQDQpRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRDQpSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSU
lJSUlJSUlJSUlJSUlJSUlJSUlJSUg0KU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1
NTU1NTU1NTU1NTU1NTU1MNClRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUDQpBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUENCkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkJCQkINC
kNDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0MNCkRERE
RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERA0KRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRQ0KRkZGRkZGRkZGRkZGRkZG
RkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRkZGRg0KR0dHR0dHR0dHR0dHR0dHR
0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHRw0KSEhISEhISEhISEhISEhISEhISEhISEhISE
hISEhISEhISEhISEhISEhISEhISEhISA0KSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUl
JSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUkNCkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpK
SkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKDQpLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS
0tLS0tLS0tLS0tLS0tLDQpMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE
xMTExMTExMTExMTA0KTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTQ0KTk5
OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTk5OTg0KT09PT09PT09P
T09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PTw0KUFBQUFBQUFBQUFBQUFBQU
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFANClFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVENClJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJSUlJ
SUlJSUlJSUlJSUlJSUlJSUlJSUlJSDQpTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NTU1NT
U1NTU1NTU1NTU1NTU1NTUw0KVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUV
FRUVFRUVFRUVFQNCg==

RequestDate:2008-09-28
ResponseDueDate:2008-10-13
FormNumber:UIB-290e
```

The State Pull Data File Model Connector expects two command line arguments:

Table 66 – JAX-WS State Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Response Collection Query.

To execute the Model Connector from the command line, type:

- *java org.uisides.client.state.StatePullClientDataFile SI|EV Data_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.state.StatePullClientDataFile SI data/StateSIPullQuery.txt*

where:

  - SI is the exchange the file is destined for

Example State Pull File

```
#SOAP Header Values
From:ST
To:Broker
PullCollection:3
StateSOAPTransactionNumber:141690


#pull query values
#mandatory field, same as From value
StatePostalCode:ST
#optional fields based on PullCollection value
StateSOAPTransactionNumber:141690
BrokerRecordEffectiveDateFrom:2010-07-13T00:00:00
BrokerRecordEffectiveDateTo:2010-07-14T00:00:00
```

7.2.4.2 Employer/TPA Model Connector – JAX-WS

This sample Model Connector demonstrates how an employer/TPA can access the UI **SIDES** Broker Web services using JAX-WS libraries.

The Model Connector was developed on JDK 1.5 (Java 5), but should also run on jdk 1.4 and Java 6.

The following main libraries are used (parts of Sun Microsystems Metro v1.4 release):

- JAX-WS RI 2.1.4.1
- JAXB RI 2.1.7.1

For convenience, this sample includes all necessary Eclipse project config files and can be imported into an existing Eclipse IDE workspace. Eclipse 3.5 (Galileo) or later is required.

7.2.4.2.1 Sample Folders and Files

Root folder: sides-employer-client-jax-ws

- ./build.xml

  Ant build file (requires Apache Ant 1.7.1 or later)
    - Run “ant generate-jaxws-client” to generate JAX-WS client Java beans from WSDL (creates src/org/uisides/client/state/generated classes)
    - Run "ant build" to compile

- ./src

  Contains:
    - Java source code
    - Log4j config file (log4j.properties)
    - employer-ws-client-config.xml – the run time settings
    - client-security-env.properties (JAX-WS security config file)

- ./lib
  Contains required library jar files. All libraries used are open-source Apache LGPL-style libraries which can be freely distributed.

- ./schemas –

  Contains UI **SIDES** XML schema files and Employer/TPA WSDL file

- ./data

  Contains sample payload xml data files for Employer/TPA Post (EmpPost.xml)

- ./bin

  Build destination folder for compiled Java class files.

7.2.4.2.2 Run Time Configuration

The Model Connector has runtime configuration parameters that allow the employer/TPA to setup its connector. The configuration is specified in a config xml file. The bean that specifies these parameters is the configParams bean. All Java Application Model Connector classes use the same configuration file, ***employer-ws-emulator-config.xml***.

Table 67 - ConfigParam options

Parameter Name	**Applies To**	**Definition**
debugLogFilePath	Post and Pull	The fully qualified location of the debug log file; it contains all the information in all the log files plus detailed information on the state of the Model Connectors workings.
resultsLogFilePath	Post and Pull	The fully qualified location of the results log file; it contains all the information with the results from the call to the Broker.
brptLogFilePath	Post and Pull	The fully qualified location of the brpt log file; it contains all the information with the results from the BRPT.
pdfFilePath	Pull	The fully qualified location of the PDF file and attachments; it will contain all the requests received in PDF form with all of the attachments decoded and stored in the same directory
writeRequestsAsPDF	Pull	A boolean value that is "true" if the responses should be printed out as the PDF and "false" otherwise.
requestFlatFilePath	Pull	The fully qualified location of the flat file containing the Request information; it will contain all the requests received in flat file format with all of the attachments still encoded
writeRequestsAsFlatFile	Pull	A boolean value that is "true" if the requests should be written in the flat file format and "false" otherwise.
pullAllFiles	Pull	A boolean value that is "true" if the connector wants the system to pull all files until the message code = 2 and "false" if the connector wants to make the call repeatedly (so as to allow the connector more control).

7.2.4.2.3 Web Services Configuration

The URL of the **SIDES** Broker Web services and **REDACTED** configuration is specified in the client-security-env.properties file.

7.2.4.2.4 Execution

This Model Connector contains four top-level Java classes with a main() method:

- EmployerPostClient

- EmployerPostClientDataFile
- EmployerPullClient
- EmployerPullClientDataFile

These files implement the employer/TPA Post and Pull operations respectively using JAX-WS.

7.2.4.2.4.1 EmployerPostClient/EmployerPullClient

The Employer Post Model Connector reads Response SOAP message payload content from XML file in the *./data* folder, sends it to the test **SIDES** Broker Web services URL, and log Broker's response to the console.

The Model Connector expects five command line arguments:

Table 68 – JAX-WS Employer Post Model Connector Command Line Arguments

Model connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
"FROM" SOAP header	This is the unique id of the Employer/TPA that is sending the file.
"TO" SOAP header	This is the unique id of the State that the file is destined for.
"EmployerTPAResponseFileGUID" SOAP header	This is the Employer/TPA Response File GUID.
The payload XML source file	The XML file that contains the payload for the call.

To execute the Model Connector from the command line, type:

- *java org.uisides.client.employer.EmployerPostClient SI|EV FROM TO EmployerTPAResponseFileGUID Payload_XML_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.employer.EmployerPostClient SI BR999999999 ST 12345678901234567890123456789012 data/EmpPost.xml*

where:

- SI is the exchange the file is destined for
- BR999999999 is the EmployerTest unique id
- ST is the unique id for StateTest UI **SIDES** endpoint
- 123456789012345678901234 is the EmployerTPAResponseFileGUID

The EmployerPullClient reads Employer pull query SOAP message payload content from XML file in the *./data* folder, sends it to the test **SIDES** Broker Web services URL, logs Broker's responses to the console, then prepares and sends Broker the acknowledgement of received pull responses.

The Model Connector expects five or six command line arguments:

Table 69 – JAX-WS Employer Pull Model Connector Command Line Arguments

Model connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
"FROM" SOAP header	This is the unique id of the Employer that is sending the pull query.
"TO" SOAP header	Always “Broker” for Pull requests
“PullCollection” SOAP header	1 for regular pull, 2 for re-pull by transmission number, 3 for re-pull by date range
The payload XML source file	The XML file that contains the payload for the call.
EmployerTPASOAPTransactionNumber (optional)	The value of the “EmployerTPASOAPTransactionNumber” SOAP header. Only required if the pullCollection parameter is 2 (re-pull by transaction number) or 3 (re-pull by date range)

To execute the Employer Pull Model Connector from the command line, type:

- *java –cp <classpath> org.uisides.client.employer.EmployerPullClient SI|EV FROM TO PullCollection Payload_XML_File_Name [EmployerTPASOAPTransactionNumber]*

Sample Model Connector arguments are:

- *java –cp <classpath> org org.uisides.client.employer.EmployerPullClient SI BR999999999 Broker 1 data/EmployerPullQuery.xml*

where:

- SI is the exchange the file is destined for
- BR999999999 is the Employer Test unique id
- Broker is the “To” value for all Pull requests
- 1 is the PullCollection value indicating a regular pull

7.2.4.2.4.2 EmployerPostClientDataFile/EmployerPullClientDataFile

These files implement the employer/TPA Post and the employer/TPA Pull/Pull Acknowledgment Web service calls respectively that read ASCII files. See Figure 1 and Figure 2.

The EmployerPostClientDateFile/EmployerPullClientDataFile Model Connectors read message payload content from ASCII flat files in the designated folder, send it to the test **SIDES** Broker Web services URL, and log Broker's response to the console.

The Employer/TPA Post Data File Model Connector expects two command line arguments:

Table 70 – JAX-WS Employer/TPA Post Data File Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Separation Response(s).

To execute the Model Connector from the command line, type:

- *java org.uisides.client.employer.EmployerPostClientDataFile SI|EV Data_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.employer.EmployerPostClientDataFile SI data/EmployerSIPost.txt*

  where:

  - SI is the exchange the file is destined for

Example Employer Response File

```
#SOAP Headers
To:ST
From:BR999999999
FileGuid:12345678901234567890123456789014

StateRequestRecordGUID:30000000000000000000000000004003
BrokerRecordTransactionNumber:2013889
SSN:560348477
ClaimEffectiveDate:2007-06-04
ClaimNumber:388620
```

```
StateEmployerAccountNbr:0065560
CorrectedEmployerName:J C Penny
CorrectedStateEmployerAccountNbr:0123456789
CorrectedFEIN:987654321
OtherSSN:660348477
ClaimantNameWorkedAsForEmployer:Andy Wilson
ClaimantJobTitle:Customer Service Associate
SeasonalEmploymentInd:N
EmployerReportedClaimantFirstDayofWork:2007-10-11
EmployerReportedClaimantLastDayofWork:2007-10-14
EffectiveSeparationDate:2007-10-14
TotalEarnedWagesNeededInd:3
TotalWeeksWorkedNeededInd:3
AverageWeeklyWage:125.00
EmployerSepReasonCode:3
ReturnToWorkInd:N
WorkingAllAvailableHoursInd:N
NotWorkingAvailableHoursReason:NotWorkingAvailableHoursReason
LaborDisputeTypeInd:L

#Remuneration
RemunerationTypeCode:3
RemunerationAmountPerPeriod:999.99
RemunerationPeriodFrequencyCode:B
DateRemunerationIssued:2007-10-15
EmployerAllocationInd:N
AllocationBeginDate:2007-10-15
AllocationEndDate:2007-10-22

AverageNumberofHoursWorkedperWeek:40
MandatoryRetirementInd:N
MandatoryPension:N
ContributoryorNotContributoryClaimantInd:N
ClaimantPensionContributionPercent:100
DischargeReasonCode:3
FinalIncidentReason:FinalIncidentReason
FinalIncidentDate:2007-10-13
ViolateCompanyPolicyInd:N
DischargePolicyAwareInd:N
DischargePolicyAwareExplanationCode:V

#PriorIncidentOccurrence
PriorIncidentDate:2007-10-10
PriorIncidentReason:None
PriorIncidentWarningInd:Y
PriorIncidentWarningDate:2007-10-10
PriorIncidentWarningDescription:Verbal

WhoDischargedName:Andy Wilson
WhoDischargedTitle:Customer Service Associate
VoluntarySepReasonCode:3
HiringAgreementChangesCode:3
HiringAgreementChangesComments:HiringAgreementChangesComments
ClaimantActionsToAvoidQuitInd:N
ContinuingWorkAvailableInd:N
PreparerTypeCode:T
```

```
PreparerCompanyName:J C Penny
PreparerTelephoneNumberPlusExt:9724312108
PreparerContactName:Ed A Jones
PreparerTitle:Project Manager
PreparerFaxNbr:9725312108
PreparerEmailAddress:edjones@jcpenneytest.com
```

The Employer/TPA Pull Data File Model Connector expects two command line arguments:

Table 71 – JAX-WS Employer/TPA Pull Model Connector Command Line Arguments

Model Connector Argument	Definition
SI | EV	This is the exchange that the file is destined for: SI – Separation Information EV – Earnings Verification
Data File	This is the fully qualified path and name of the data file that contains the flat file structure of the Request Collection Query.

To execute the Model Connector from the command line, type:

- *java org.uisides.client.state.EmployerPullClientDataFile SI|EV Data_File_Name*

Sample Model Connector arguments are:

- *java org.uisides.client.state.EmployerPullClientDataFile SI data/StateSIPullQuery.txt*

where:

  - SI is the exchange the file is destined for

Example Employer/TPA Pull File

```
#SOAP Header Values
From:BR000000003
To:Broker
PullCollection:3
EmployerTPASOAPTransactionNumber:141690

#pull query values
#mandatory field, same as From value
UniqueID:BR000000003
#optional fields based on PullCollection value
EmployerTPASOAPTransactionNumber:141690
BrokerRecordEffectiveDateFrom:2010-07-13T00:00:00
BrokerRecordEffectiveDateTo:2010-07-14T00:00:00
```

7.2.5 BRPT – Business Rule Processor Tool

There are two key data checks that enforce the quality of data transmitted in the **SIDES** system: XSDs and Business Rules. The XSD validation is automatically provided by the J2EE or .Net parser technology as part of the XML file generation and consumption process. However, the **SIDES** data-related business rules require implementation via programming code and it is vital that all connectors and the Broker implement these business rules precisely and correctly. The purpose of the Business Rule Processor Tool is twofold:

- The Business Rule Processor Tool ensures a standard implementation and interpretation of the **SIDES** business rules, not just for the current **SIDES** members, but new participants in the future. For future participants, demonstrating the correct implementation of the business rules is part of certifying their readiness to join **SIDES** production operations.
- The Business Rule Processor Tool permits an Endpoint to test their business rule programming code prior to connecting to the Broker. Uncovering errors during unit test will be much less costly than discovering them during testing with the Broker. Technically, the tool accepts a Data Transfer Object ("DTO") or XML input and returns the errors and associated codes that the Broker will provide without the need to connect to the Broker. The tool has been developed in both .Net (C#) and Java 2 Platform, Enterprise Edition ("J2EE") allowing all connectors to take advantage of it.

The Business Rule Processor Tool is a self contained jar file or a DLL (depending on the language) that can be included in the connectors' software project. It has exposed interfaces or data transfer objects (DTOs) that a connector can invoke from within their code to test the Broker business rules.

7.2.5.1 BRPT Interfaces

The Business Rule Processing Rule (BRPT) is accessed through twelve different methods, 3 for Separation Information request, 3 for Separation Information response, 3 for Earnings Verification request and 3 for Earnings Verification response, per language (.Net (C#) and Java).

The first way to access the BRPT is with an XML file. The connector can either have their system write the XML to a file or the connector can build the file by hand with data out of its system to verify the data's integrity.

The second way to access the BRPT is with an XML Stream. This method allows the connector to build the XML using their system in a form that it can use to send to the Broker but allows the testing of the data to occur without having to have a working connection to the Broker.

The third way to access the BRPT is with a Data Transfer Object (DTO). This method allows the connector to populate a DTO and verify the data meets **SIDES** specifications.

7.2.5.1.1 Java XML File

Table 72 – BRPT Java XML File

Method Name	Parameters	Description
`processEarningVerificationRequests`	`String requestsFilePath`	Process the Earnings Verification Requests contained in the File at requestFilePath.
`processEarningsVerificationResponses`	`String responsesFilePath`	Process the Earnings Verification Responses contained in the File at responseFilePath.
`processSeparationRequests`	`String requestsFilePath`	Process the Separation Requests contained in the File at requestFilePath.
`processSeparationResponses`	`String responsesFilePath`	Process the Separation Responses contained in the File at responseFilePath.

7.2.5.1.2 Java XML Construct

Table 73 – BRPT Java XML Construct

Method Name	Parameters	Description
`processEarningVerificationRequests`	`XMLStreamReader requestsStreamReader`	Process the Earnings Verification Requests contained within the XMLStreamReader requestStreamReader
`processEarningsVerificationResponses`	`XMLStreamReader responsesStreamReader`	Process the Earnings Verification Responses contained within XMLStreamReader responsesStreamReader.
`processSeparationRequests`	`XMLStreamReader requestsStreamReader`	Process the Separation Requests contained within the

Method Name	Parameters	Description
		XMLStreamReader requestStreamReader.
`processSeparationResponses`	`XMLStreamReader responsesStreamReader`	Process the Separation Responses contained within the XMLStreamReader responsesStreamReader.

7.2.5.1.3 Java Data Transfer Object

Table 74 – BRPT Java Data Transfer Object

Method Name	Parameters	Description
`processEarningVerificationRequests`	`StateEarningsVerificationRequestCollection requests`	Process the Earnings Verification Requests contained within the requests Data Transfer Object.
`processEarningsVerificationResponses`	`EmployerTPAEarningsVerificationCollection responses`	Process the Earnings Verification Responses contained within the responses Data Transfer Object.
`processSeparationRequests`	`StateSeparationRequestCollection requests`	Process the Separation Requests contained within the requests Data Transfer Object.
`processSeparationResponses`	`EmployerTPASeparationResponseCollection responses`	Process the Separation Responses contained within the responses Data Transfer Object.

7.2.5.1.4 .Net (C#) XML File

Table 75 – BRPT .Net (C#) XML File

Method Name	Parameters	Description
`processEarningVerificationRequests`	`String requestsFilePath`	Process the Earnings Verification Requests contained in the File at requestFilePath.
`processEarningsVerificationResponses`	`String responsesFilePath`	Process the Earnings Verification Responses contained in the File at responseFilePath.
`processSeparationRequests`	`String requestsFilePath`	Process the Separation Requests contained in the File at requestFilePath.
`processSeparationResponses`	`String responsesFilePath`	Process the Separation Responses contained in the File at responseFilePath.

7.2.5.1.5 .Net (C#) XML Construct

Table 76 – BRPT .Net(C#) XML Construct

Method Name	Parameters	Description
`processEarningVerificationRequests`	`XMLReader requestsStreamReader`	Process the Earnings Verification Requests contained within the XMLReader requestStreamReader.
`processEarningsVerificationResponses`	`XMLReader responsesStreamReader`	Process the Earnings Verification Responses contained within the XMLReader responsesReader.
`processSeparationRequests`	`XMLReader requestsStreamReader`	Process the Separation Requests contained within the

Method Name	Parameters	Description
		XMLReader requestStreamReader.
processSeparationResponses	XMLReader responsesStreamReader	Process the Separation Responses contained within the XMLReader responsesReader.

7.2.5.1.6 .Net (C#) Data Transfer Object

Table 77 – BRPT .Net (C#) Data Transfer Object

Method Name	Parameters	Description
processEarningVerificationRequests	StateEarningsVerificationRequestCollection requests	Process the Earnings Verification Requests contained within the requests Data Transfer Object.
processEarningsVerificationResponses	EmployerTPAEarningsVerificationResponseCollection responses	Process the Earnings Verification Responses contained within the responses Data Transfer Object.
processSeparationRequests	StateSeparationRequestCollection requests	Process the Separation Requests contained within the requests Data Transfer Object.
processSeparationResponses	EmployerTPASeparationResponseCollection responses	Process the Separation Responses contained within the responses Data Transfer Object.

7.2.5.2 Return from Business Rules Processing Tool

The return from a call to the BRPT is an object type that contains the status information and Error Codes that are passed back to the state and employer/TPA from the call to a Broker.

For the Earnings Verification calls, it simulates the XML in the form of StateEarningsVerificationRequestCollectionAcknowledgement and EmployerTPAEarningsVerificationResponseCollectionAcknowledgement defined in the Earnings Verification Request and Earnings Verification Response XSD, except it has already been transformed into an object.

For the Separation Information calls, it simulates the XML in the form of StateSeparationRequestCollectionAcknowledgement and EmployerTPASeparationResponseCollectionAcknowledgement defined in the Separation Request and Separation Response XSD, except it has already been transformed into an object.

7.2.5.3 Example Invocation of the Business Rules Processing Tool

The following code snippets give examples of the invocation of the Separation Information code once the libraries are included in the project.

7.2.5.3.1 Java Example Invocation

7.2.5.3.1.1 State Java Example Invocation

```
// State
RequestBRProcessorImpl client = RequestBRProcessorImpl.getInstance();

// State Invocation with File Name s
StateSeparationRequestCollectionAcknowledgement requestDto =
client.processSeparationRequests((String) s);

// State Invocation with XML Stream Reader xsr
StateSeparationRequestCollectionAcknowledgement responseDto =
client.processSeparationRequests(xsr);


// State Invocation with DTO requests
StateSeparationRequestCollectionAcknowledgement responseDto =
client.processSeparationRequests(requests);
```

7.2.5.3.1.2 Employer/TPA Java Example Invocation

```
// Employer/TPA
ResponseBRProcessorImpl client = ResponseBRProcessorImpl.getInstance();

// Employer/TPA Invocation with File Name s
EmployerTPASeparationResponseCollectionAcknowledgement responseDto =
client.processSeparationResponses((String) s);
// Employer/TPA Invocation with XML Stream Reader xsr

EmployerTPASeparationResponseCollectionAcknowledgement responseDto =
client.processSeparationResponses(xsr);

// Employer/TPA Invocation with DTO responses
EmployerTPASeparationResponseCollectionAcknowledgement responseDto =
client.processSeparationResponses(responses);
```

7.2.5.3.2 .Net Example Invocation

7.2.5.3.2.1 State .Net Example Invocation

```
// State
RequestBRProcessorImpl me = new RequestBRProcessorImpl();

// State Invocation with File Name fileNameObj
StateSeparationRequestCollectionAcknowledgement ack =
me.processSeparationRequests(fileNameObj.ToString());

// State Invocation with XML Reader xr
StateSeparationRequestCollectionAcknowledgement ack =
me.processSeparationRequests(xr);

// State Invocation with DTO requests
StateSeparationRequestCollectionAcknowledgement ack =
me.processSeparationRequests(requests);
```

7.2.5.3.2.2 Employer/TPA .Net Example Invocation

```
// Employer/TPA
ResponseBRProcessorImpl me = new ResponseBRProcessorImpl();

// Employer/TPA Invocation with File Name fileNameObj
EmployerTPASeparationResponseCollectionAcknowledgement ack =
me.processSeparationResponses(fileNameObj.ToString());

// Employer/TPA Invocation with XML Reader xr
EmployerTPASeparationResponseCollectionAcknowledgement ack =
me.processSeparationResponses(xr);

// Employer/TPA Invocation with DTO responses
EmployerTPASeparationResponseCollectionAcknowledgement ack =
me.processSeparationResponses(responses);
```

8 F – CONNECT WITH THE CENTRAL BROKER: CERTIFYING CONNECTOR SOFTWARE

This section provides details and discussion on the final testing process where the connector must certify it is ready to interface with the production **Central Broker**. This is a critical section as the connector software must pass the certification tests before being allowed to enter production with **SIDES**.

8.1 Certification

To utilize the **SIDES** **Central Broker** platform, the state, employer or TPA connector software must meet a set of agreed upon business rule validation requirements. A key requirement of **SIDES** is to ensure the quality and integrity of data exchanged between connectors. To meet this requirement, the **Central Broker** performs edit validation and business rule validation on the data it receives, and connectors must validate the data locally prior to submitting. This section describes the process **SIDES** will use to certify that a connector has correctly implemented its validations prior to enabling access to the production UI **SIDES** **Central Broker**.

NOTE: In order to perform the **connector** certification test, a **connector** must ensure that their software is developed with the capability to allow the injection of Certification data. See Section 9.2.2.1

A connector may use different technologies and programming languages to create their client program, therefore the certification process does not inspect the client source code or design, but relies instead on a set of input data and expected outcomes to test compliance. The only client design features required for connector certification is the client's ability to load and use test data from XML files (provided by the **Central Broker**), and the ability to produce a text log file or database records listing any validation errors detected in supplied data.

Connectors will be provided with test XML files and a spreadsheet listing error codes and associated test files; there is one set each for state and employer/TPA connectors. XML file sets will contain both State Request and Employer Response data files. Both valid and invalid data files will be included.

The connector certification process consists of two steps. First, the connector performs preliminary certification testing. During the preliminary certification of the connector, a state or employer/TPA representative will use their connector to submit **SIDES**-provided test data to the **Central Broker** test environment. After processing each input file, the certifier will inspect client logs to validate that all expected validation errors were caught, and the certifier will validate that clean data was successfully passed to the **Central Broker**. The tester also needs to ensure that all **Central Broker** message codes or business rule error codes are processed appropriately by the connector software or back-end system. Validation results are recorded in the provided spreadsheet.

The final certification test is initiated after the connector completes their preliminary certification test. The **SIDES** **Broker** Administrator will review the connector’s test results spreadsheet and all certification data files will be submitted to the SIDES **Central Broker** test environment. The SIDES technical team will review the **Central Broker** reports to ensure expected results were achieved and if so, the connector will be certified.

States, employers, and TPAs can use tools provided by the SIDES technical team to prepare for certification. The tools include Model Connectors and the Business Rules Processing Tool (BRPT), which provides a reference implementation for client-side validation. The BRPT tool is implemented in both Java and C#. The BRPT graphical interface takes an XML data file as input and validates it against the **Central Broker** business rules, reporting any errors. Connector client developers can incorporate the BRPT source code as a data validation module into their client program.

The Model Connectors built by the SIDES technical team are implemented in both Java and .Net. The Model Connectors allow a connector to act as the opposite endpoint. As a result, the state, employer or TPA may test their own connector software without having to rely on an outside party to complete the round-trip exchange of information.

8.1.1 Certification Information

Connector certification is a required step that must be performed prior to production operations or after any major change to the connector system. It is important that the connector software be properly vetted so that valuable production time is not spent on items that could have been prevented during connector testing.

Certification of the connector is achieved when:

- The expected results of the test data files match the actual test results from the connector.
- The connecter back-end processing is verified to handle / process all message codes or business rule error codes.
- The connector handles duplicate processing.

The following process needs to be followed before a connector will be allowed to join SIDES in production.

- Step 1 - Download the test suite of XML files and spreadsheets from the SIDES Website.
- Step 2 – Conduct preliminary connector certification testing.
  - Step 2.1 – Run XML files through the connector.
  - Step 2.2 – Compare the results obtained from your test system with the expected results.
  - Step 2.3 - Fill in the spreadsheet with the results of the test.
- Step 3 - Send the completed document to the SIDES Business Manager.
- Step 4 –Conduct final connector certification test.

8.1.1.1 Step 1 - Download Test Suite

The test suite can be found on the **SIDES** Website at http://sides.itsc.org. Navigate to the Connector Certification section to download the following files from the folder that contains the certification files for the exchange you want to be certified for:

- Client_Certification_State_worksheet.xls
- Client_Certification_Employer-TPA_worksheet.xls
- Certification_Data.zip

The Client_Certification_State_worksheet.xls file (for state connectors) and Client_Certification_Employer-TPA_worksheet.xls file (for employer/TPA connectors) list both valid and invalid files to be used during client certification. The spreadsheets also list test files for duplicate processing, post message codes, and pull message codes.

For certification of state connectors, states will test business rule error codes associated with valid request files and invalid request files. Valid request files are expected to be processed by the connector software without business rule errors and to be successfully posted to the **Central Broker.** Corresponding valid response files will be pulled from the **Central Broker** to test the Pull message codes. Invalid requests are expected to be rejected by the connector software, trapping specified business rule errors. State connectors must self-certify duplicate response processing, post / pull message code handling, and attachment processing.

For certification of employer or TPA connectors, employers or TPAs will test business rule error codes associated with valid response files and invalid response files. Valid response files are expected to be processed by the connector software without business rule errors and to be successfully posted to the **Central Broker.** Corresponding valid request files will be pulled from the **Central Broker** to test the Pull message codes. Invalid responses are expected to be rejected by the connector software, trapping specified business rule errors. Employer or TPA connectors must self-certify duplicate request processing, post / pull message code handling, and attachment processing.

The state, employer or TPA tester will fill out the highlighted rows in both tabs of their worksheet and return the worksheet to the **SIDES** Business Manager.

8.1.1.1.1 Data Files

The individual data files are located inside the 'XML datasets' directory structure when the Test Suite files are unzipped.

For Earnings Verification, within the 'XML datasets' directory are sub-directories that are broken into logical groups based on the ClaimantEmployerWorkRelationship (ER-15), the EmployerEarningsCode (ER-16) and the Status Codes requested. Within each sub-directory are the data files. Data files may contain from 1 to 5 requests and associated responses. Also, there is a Business Rules Error folder which test all business rules corresponding and a Boundary Cases folder that tests the limits of each field.

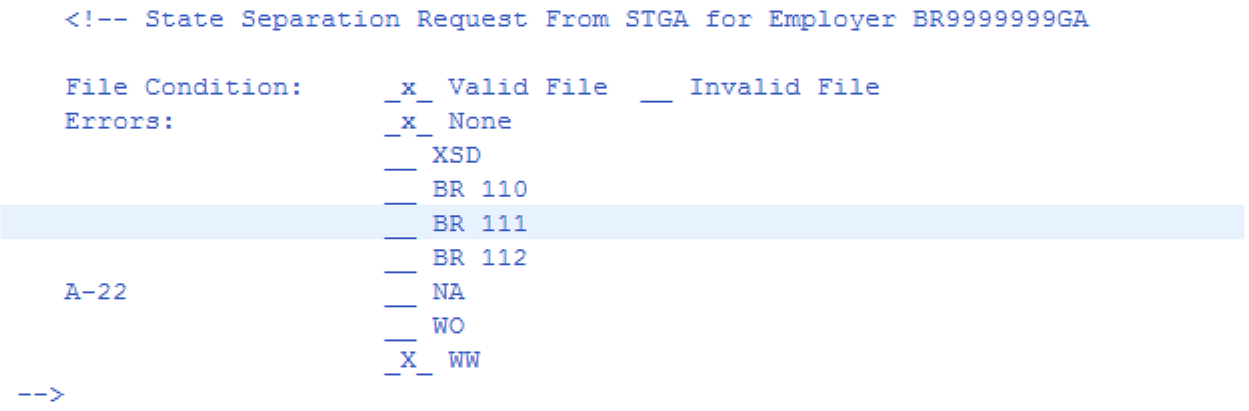
For Separation Information, within the ‘XML datasets’ directory are sub-directories that are broken into logic groups based on separation information reason codes (reason for separation). Within each sub-directory are the data files. Data files may contain from 4 to 12 requests and associated responses. These files test all business rules corresponding to the separation information reason code.

A connector can manipulate the data file to ensure the data can be processed through the connector software. For example, a state connector may be unable to process the provided request data with the given StateEmployerAccountNbr and the connector may need to modify the StateEmployerAccountNbr to successfully post the request to the **Central Broker**. Similarly, a state connector may need to use their own StateRequestRecordGUID rather than the one provided in the certification test data. When a connector prepares a response file, the BrokerRecordTransactionNumber must be updated to match the BrokerRecordTransactionNumber generated by the Central Broker. The **SIDES** team suggests that connectors minimize changes to the certification test data to ensure the integrity of the certification test.

The **SIDES** **Central Broker** requirements list a set of business rules, which State Request and Employer/TPA Response data must satisfy prior to being transmitted by a connector. Each XML file in the test suite contains a header that details the business rules being checked within the test file.

Separation Request File Header

A sample header file for a state separation request XML data file follows below:



```
    <!-- State Separation Request From STGA for Employer BR9999999GA

    File Condition:     _x_ Valid File   __ Invalid File
    Errors:             _x_ None
                        __ XSD
                        __ BR 110
                        __ BR 111
                        __ BR 112
    A-22                __ NA
                        __ WO
                        _X_ WW
 -->
```

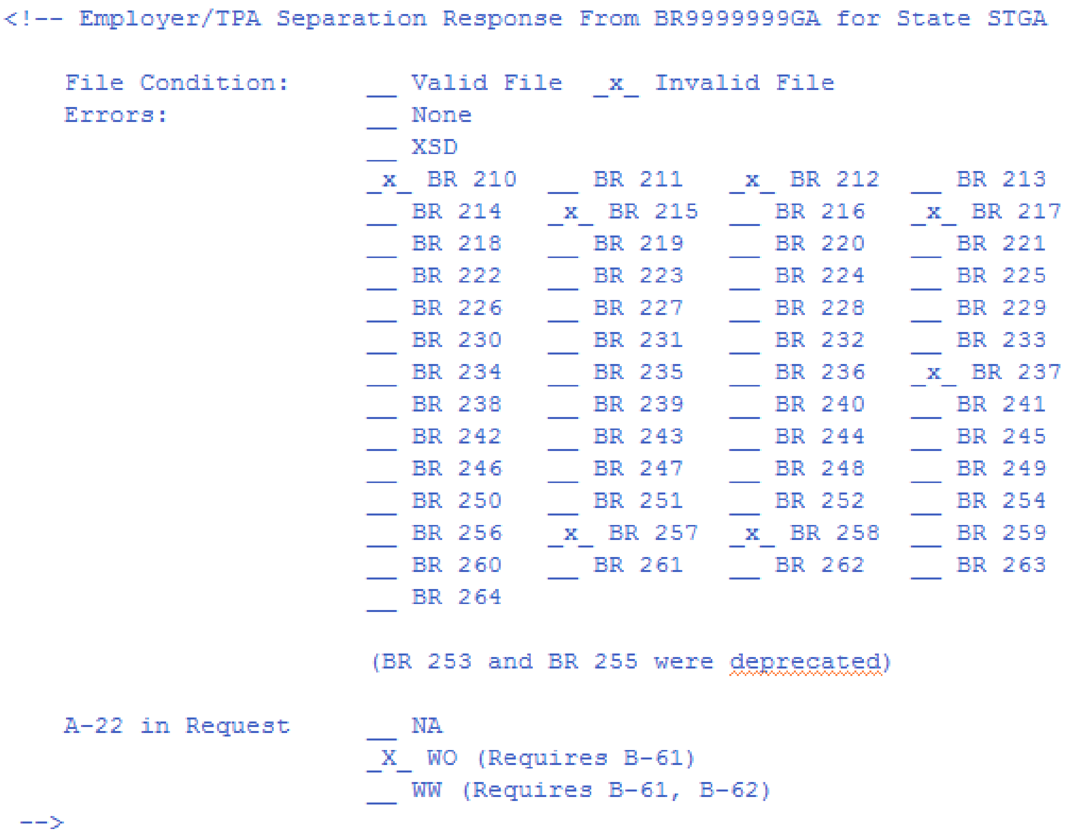
Table 78 – Separation Request File Header

Header Category	Results
File Condition	This indicates whether this is a valid file or an invalid file. This will tell the tester if the expected result is a successful pass through or a failure and errors should be logged.

Header Category	**Results**
Errors	This indicates whether the file has no errors, an XSD error or a business rule error. Business rules errors are listed individually. More than one business rules can be tested in the file.
A-22	This indicates the value of A-22 from the Separation Information Standard Format in the request. This field is singled out as a critical field because it will influence the business rules on both the request and response.

Separation Response File Header

A sample header file for an employer / TPA separation response XML data file follows below:



```
<!-- Employer/TPA Separation Response From BR9999999GA for State STGA

    File Condition:      __ Valid File  _x_ Invalid File
    Errors:              __ None
                         __ XSD
                         _x_ BR 210  __ BR 211  _x_ BR 212  __ BR 213
                         __ BR 214  _x_ BR 215  __ BR 216  _x_ BR 217
                         __ BR 218  __ BR 219  __ BR 220  __ BR 221
                         __ BR 222  __ BR 223  __ BR 224  __ BR 225
                         __ BR 226  __ BR 227  __ BR 228  __ BR 229
                         __ BR 230  __ BR 231  __ BR 232  __ BR 233
                         __ BR 234  __ BR 235  __ BR 236  _x_ BR 237
                         __ BR 238  __ BR 239  __ BR 240  __ BR 241
                         __ BR 242  __ BR 243  __ BR 244  __ BR 245
                         __ BR 246  __ BR 247  __ BR 248  __ BR 249
                         __ BR 250  __ BR 251  __ BR 252  __ BR 254
                         __ BR 256  _x_ BR 257  _x_ BR 258  __ BR 259
                         __ BR 260  __ BR 261  __ BR 262  __ BR 263
                         __ BR 264

                         (BR 253 and BR 255 were deprecated)

    A-22 in Request      __ NA
                         _X_ WO (Requires B-61)
                         __ WW (Requires B-61, B-62)
 -->
```

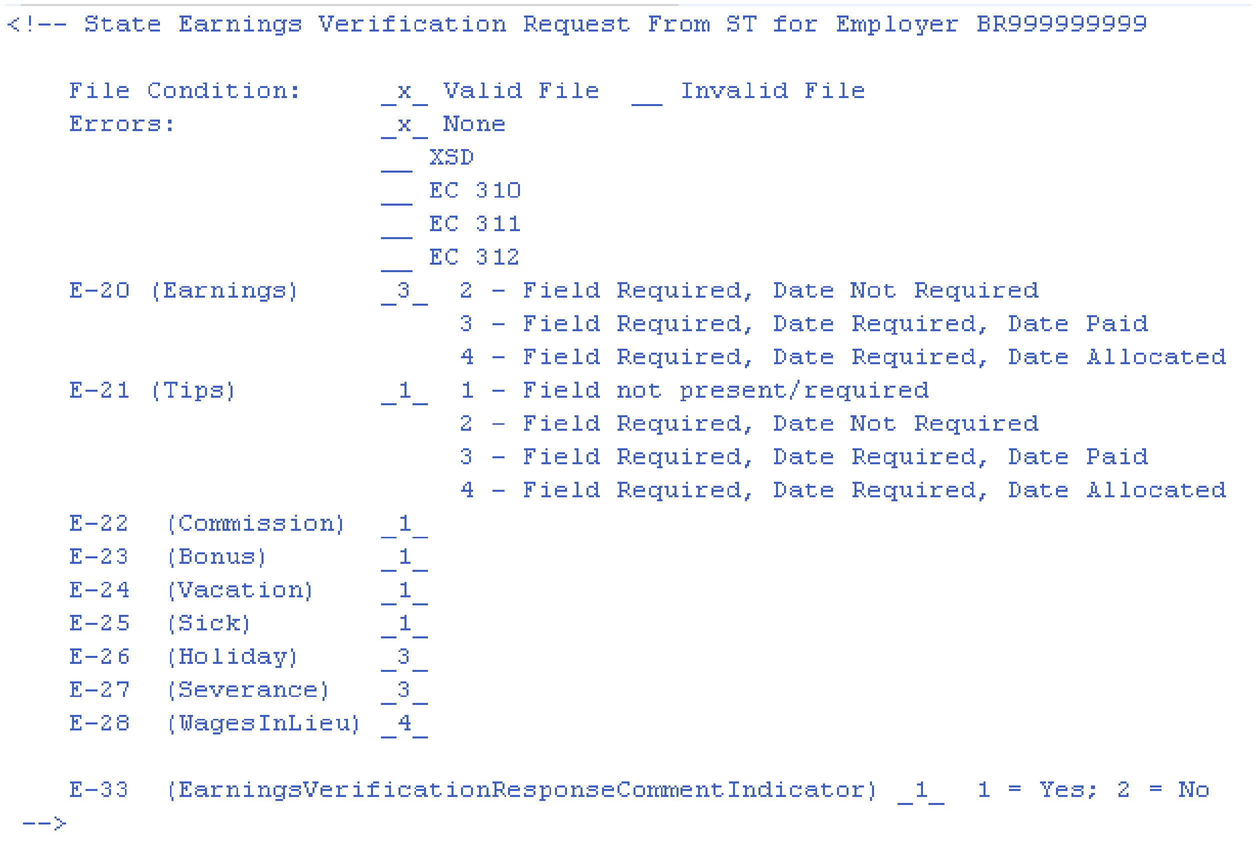
Table 79 – Separation Response File Header

Header Category	**Results**
File Condition	This indicates whether this is a valid file or an invalid file. This will tell the tester if the expected result is a successful pass through or a failure and errors

Header Category	Results
	should be logged.
Errors	This indicates whether the file has no errors, an XSD error or a business rule error. Business rules errors are listed individually. More than one business rules can be tested in the file.
A-22 in Request	This indicates the value of A-22 from the Separation Information Standard Format in the request. This field is singled out as a critical field because it will influence the business rules on both the request and response.

Earnings Verification Request File Header

A sample header file for a state earnings verification request XML data file follows below:



```
<!-- State Earnings Verification Request From ST for Employer BR999999999

    File Condition:      _x_ Valid File  __ Invalid File
    Errors:              _x_ None
                         __ XSD
                         __ EC 310
                         __ EC 311
                         __ EC 312
    E-20 (Earnings)      _3_  2 - Field Required, Date Not Required
                              3 - Field Required, Date Required, Date Paid
                              4 - Field Required, Date Required, Date Allocated
    E-21 (Tips)          _1_  1 - Field not present/required
                              2 - Field Required, Date Not Required
                              3 - Field Required, Date Required, Date Paid
                              4 - Field Required, Date Required, Date Allocated
    E-22  (Commission)   _1_
    E-23  (Bonus)        _1_
    E-24  (Vacation)     _1_
    E-25  (Sick)         _1_
    E-26  (Holiday)      _3_
    E-27  (Severance)    _3_
    E-28  (WagesInLieu)  _4_

    E-33  (EarningsVerificationResponseCommentIndicator) _1_  1 = Yes; 2 = No
 -->
```

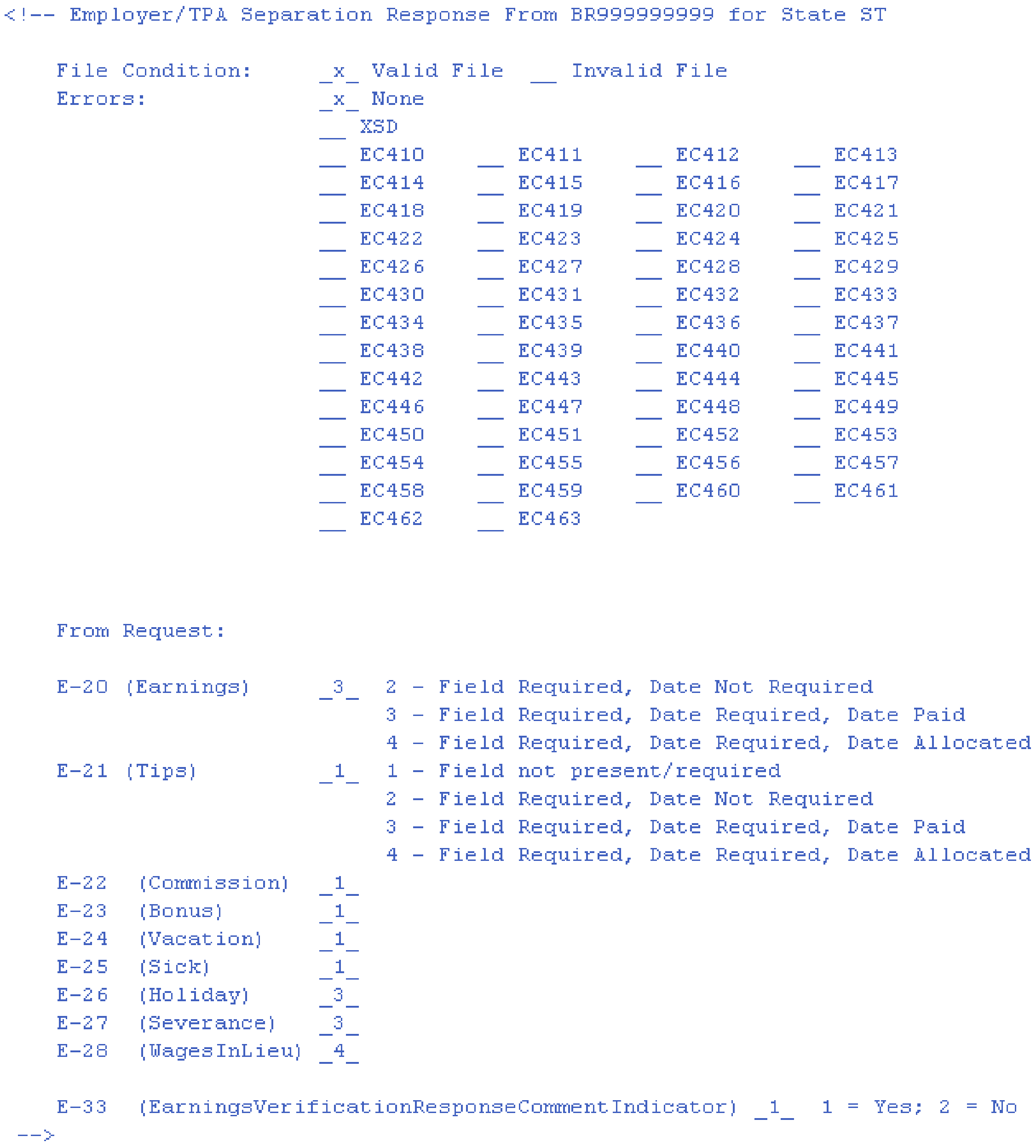
Table 80 – Earnings Verification Request File Header

Header Category	Results
File Condition	This indicates whether this is a valid file or an invalid file. This will tell the

Header Category	Results
	tester if the expected result is a successful pass through or a failure and errors should be logged.
Errors	This indicates whether the file has no errors, an XSD error or a business rule error. Business rules errors are listed individually. More than one business rules can be tested in the file.
E-20 thru E-28	This indicates the value of the Status Code from the Earnings Verification Request fields E-20 thru E-28. These fields determine the Earnings Verification Response fields that must be included in the Response.
E-33	This indicates the value of the Earnings Verification Response Comments Indicator which tell the Employer/TPA whether they can have a comment field to present further information.

Earnings Verification Response File Header

A sample header file for an employer / TPA earnings verification response XML data file follows below:



```
<!-- Employer/TPA Separation Response From BR999999999 for State ST

    File Condition:     _x_ Valid File   __ Invalid File
    Errors:             _x_ None
                        __  XSD
                        __  EC410    __  EC411    __  EC412    __  EC413
                        __  EC414    __  EC415    __  EC416    __  EC417
                        __  EC418    __  EC419    __  EC420    __  EC421
                        __  EC422    __  EC423    __  EC424    __  EC425
                        __  EC426    __  EC427    __  EC428    __  EC429
                        __  EC430    __  EC431    __  EC432    __  EC433
                        __  EC434    __  EC435    __  EC436    __  EC437
                        __  EC438    __  EC439    __  EC440    __  EC441
                        __  EC442    __  EC443    __  EC444    __  EC445
                        __  EC446    __  EC447    __  EC448    __  EC449
                        __  EC450    __  EC451    __  EC452    __  EC453
                        __  EC454    __  EC455    __  EC456    __  EC457
                        __  EC458    __  EC459    __  EC460    __  EC461
                        __  EC462    __  EC463


    From Request:

    E-20 (Earnings)     _3_  2 - Field Required, Date Not Required
                             3 - Field Required, Date Required, Date Paid
                             4 - Field Required, Date Required, Date Allocated
    E-21 (Tips)         _1_  1 - Field not present/required
                             2 - Field Required, Date Not Required
                             3 - Field Required, Date Required, Date Paid
                             4 - Field Required, Date Required, Date Allocated
    E-22  (Commission)  _1_
    E-23  (Bonus)       _1_
    E-24  (Vacation)    _1_
    E-25  (Sick)        _1_
    E-26  (Holiday)     _3_
    E-27  (Severance)   _3_
    E-28  (WagesInLieu) _4_

    E-33  (EarningsVerificationResponseCommentIndicator) _1_  1 = Yes; 2 = No
 -->
```

Table 81 – Separation Response File Header

Header Category	Results
File Condition	This indicates whether this is a valid file or an invalid file. This will tell the tester if the expected result is a successful pass through or a failure and errors should be logged.
Errors	This indicates whether the file has no errors, an XSD error or a business rule error. Business rules errors are listed individually. More than one business

Header Category	Results
	rules can be tested in the file.
E-20 thru E-28	This indicates the value of the Status Code from the Earnings Verification Request fields E-20 thru E-28. These fields determine the Earnings Verification Response fields that must be included in the Response.
E-33	This indicates the value of the Earnings Verification Response Comments Indicator which tell the Employer/TPA whether they can have a comment field to present further information.

8.1.1.1.2 Business Rule Validation

The certification data files will test the entire suite of SIDES business rules for either the state connector, or employer or TPA connector, which are specified below. When the connector software is successfully tested against each business rule, the connector will then be certified for production operations.

Note: In the headers of the XML certification test data files for Separation Information, BR### will return the Error Code ###. For example, BR101will return Error Code 101 in the acknowledgement.

Example – State Separation Request:

Table 82 – State Request Business Rules/Error Codes

Business Rules	Error Code
BR101	101
BR102	102
BR110	103

Example – Employer/TPA Separation Response:

Table 83 – Employer/TPA Response Business Rules/Error Codes

Business Rule	Error Code
BR201	201
BR202	202

Business Rule	Error Code
BR210	210

Note: In the headers of the XML certification test data files for Earnings Verification, EC### will return the Error Code ###. For example, EC301will return Error Code 301 in the acknowledgement.

Example – State Earnings Verification Request:

Table 84 – State Request Business Rules/Error Codes

Business Rules	Error Code
EC301	301
EC310	303

Example – Employer/TPA Earnings Verification Response:

Table 85 – Employer/TPA Response Business Rules/Error Codes

Business Rule	Error Code
EC401	401
EC410	410

8.1.1.2 Step 2 - Conduct Preliminary Connector Certification Testing

The bulk of the connector certification process takes place during the preliminary connector testing step. During preliminary connector certification testing, the state, employer or TPA will use their connector software and a Model Connector (one provided by the **SIDES** technical support team or their own), and a test account to submit each certification test file through their system to the **SIDES** **Central Broker** test environment.

For each certification test file that is submitted, the expected results (documented in the test file headers) must be compared to the results returned from the **SIDES** **Central Broker**. Similarly, all **Central Broker** message codes or error codes must be handled successfully by the connector software or back-end system. Connectors must test duplicate processing and post / pull message codes.

In the provided spreadsheets, test results must be documented. Specifically all highlighted cells in both tabs of the worksheet must be filled out. If there are discrepancies such as the Enter Post Message Code column is not equal to 1 (successfully posted to the Central Broker), the connector software engineers must investigate the issue and perform software remediation as needed.

This process is completed for all certification test data files until all connector software results match the expected results.

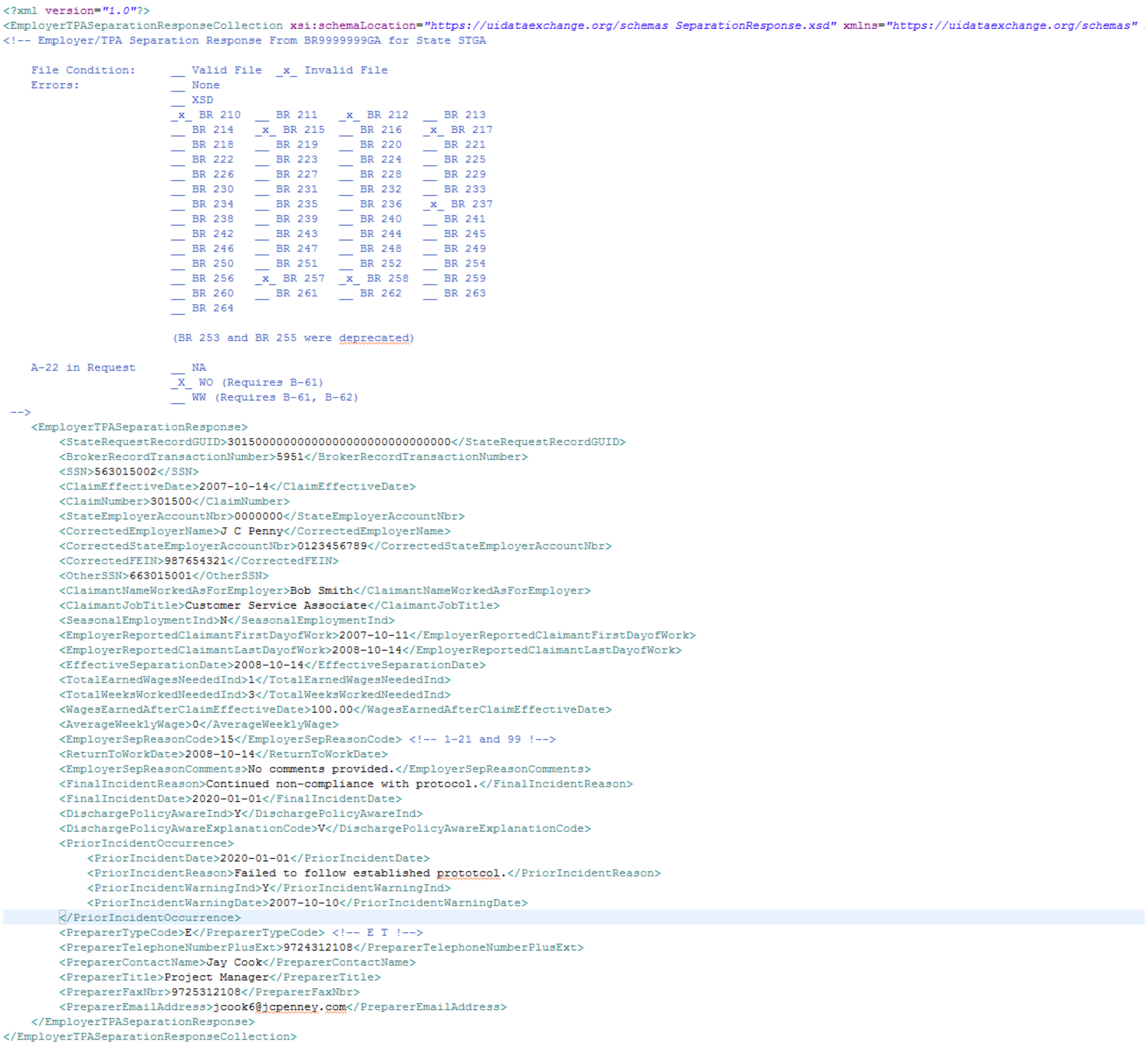
8.1.1.2.1 Step 2.1 – Run XML Files through the Connector

Each file within the test suite is an XML file that tests zero, one or more test conditions. The valid files should pass through the connector software business rules and be passed on to the **Central Broker**. The invalid files must be trapped by the connector software prior to delivery to the **Central Broker**.

The injection point into the connector software is crucial to being able to perform this step in the process. During design and development, the connector must have implemented an XML data injection point that will accept XML files in the **SIDES** Standard format into the application processing flow. The injection point to Post data to the **Central Broker** should be placed in the connector software somewhere before the outgoing business rules are checked and the SOAP message in completed. See Section 9.2.2.1

Example

An employer or TPA has chosen to test with the Separation Information response file Code_3_15_Data_Set_2_Response.xml. This file is an invalid file and tests the following business rules: BR210, BR212, BR215, BR217, BR237, BR257, and BR258. Upon execution of the connector, the data file is injected to Post a separation response to the **Central Broker**. The file should produce errors and not be transmitted to the **Central Broker**.



```
<?xml version="1.0"?>
<EmployerTPASeparationResponseCollection xsi:schemaLocation="https://uidataexchange.org/schemas SeparationResponse.xsd" xmlns="https://uidataexchange.org/schemas"
<!-- Employer/TPA Separation Response From BR9999999GA for State STGA

    File Condition:     __ Valid File  _x_ Invalid File
    Errors:             __ None
                        __ XSD
                        _x_ BR 210  __ BR 211   _x_ BR 212  __ BR 213
                        __ BR 214  _x_ BR 215  __ BR 216  _x_ BR 217
                        __ BR 218  __ BR 219  __ BR 220  __ BR 221
                        __ BR 222  __ BR 223  __ BR 224  __ BR 225
                        __ BR 226  __ BR 227  __ BR 228  __ BR 229
                        __ BR 230  __ BR 231  __ BR 232  __ BR 233
                        __ BR 234  __ BR 235  __ BR 236  _x_ BR 237
                        __ BR 238  __ BR 239  __ BR 240  __ BR 241
                        __ BR 242  __ BR 243  __ BR 244  __ BR 245
                        __ BR 246  __ BR 247  __ BR 248  __ BR 249
                        __ BR 250  __ BR 251  __ BR 252  __ BR 254
                        __ BR 256  _x_ BR 257  _x_ BR 258  __ BR 259
                        __ BR 260  __ BR 261  __ BR 262  __ BR 263
                        __ BR 264

                        (BR 253 and BR 255 were deprecated)

    A-22 in Request     __ NA
                        _X_ WO (Requires B-61)
                        __ WW (Requires B-61, B-62)
 -->
    <EmployerTPASeparationResponse>
        <StateRequestRecordGUID>30150000000000000000000000000000</StateRequestRecordGUID>
        <BrokerRecordTransactionNumber>5951</BrokerRecordTransactionNumber>
        <SSN>563015002</SSN>
        <ClaimEffectiveDate>2007-10-14</ClaimEffectiveDate>
        <ClaimNumber>301500</ClaimNumber>
        <StateEmployerAccountNbr>0000000</StateEmployerAccountNbr>
        <CorrectedEmployerName>J C Penny</CorrectedEmployerName>
        <CorrectedStateEmployerAccountNbr>0123456789</CorrectedStateEmployerAccountNbr>
        <CorrectedFEIN>987654321</CorrectedFEIN>
        <OtherSSN>663015001</OtherSSN>
        <ClaimantNameWorkedAsForEmployer>Bob Smith</ClaimantNameWorkedAsForEmployer>
        <ClaimantJobTitle>Customer Service Associate</ClaimantJobTitle>
        <SeasonalEmploymentInd>N</SeasonalEmploymentInd>
        <EmployerReportedClaimantFirstDayofWork>2007-10-11</EmployerReportedClaimantFirstDayofWork>
        <EmployerReportedClaimantLastDayofWork>2008-10-14</EmployerReportedClaimantLastDayofWork>
        <EffectiveSeparationDate>2008-10-14</EffectiveSeparationDate>
        <TotalEarnedWagesNeededInd>1</TotalEarnedWagesNeededInd>
        <TotalWeeksWorkedNeededInd>3</TotalWeeksWorkedNeededInd>
        <WagesEarnedAfterClaimEffectiveDate>100.00</WagesEarnedAfterClaimEffectiveDate>
        <AverageWeeklyWage>0</AverageWeeklyWage>
        <EmployerSepReasonCode>15</EmployerSepReasonCode> <!-- 1-21 and 99 !-->
        <ReturnToWorkDate>2008-10-14</ReturnToWorkDate>
        <EmployerSepReasonComments>No comments provided.</EmployerSepReasonComments>
        <FinalIncidentReason>Continued non-compliance with protocol.</FinalIncidentReason>
        <FinalIncidentDate>2020-01-01</FinalIncidentDate>
        <DischargePolicyAwareInd>Y</DischargePolicyAwareInd>
        <DischargePolicyAwareExplanationCode>V</DischargePolicyAwareExplanationCode>
        <PriorIncidentOccurrence>
            <PriorIncidentDate>2020-01-01</PriorIncidentDate>
            <PriorIncidentReason>Failed to follow established prototcol.</PriorIncidentReason>
            <PriorIncidentWarningInd>Y</PriorIncidentWarningInd>
            <PriorIncidentWarningDate>2007-10-10</PriorIncidentWarningDate>
        </PriorIncidentOccurrence>
        <PreparerTypeCode>E</PreparerTypeCode> <!-- E T !-->
        <PreparerTelephoneNumberPlusExt>9724312108</PreparerTelephoneNumberPlusExt>
        <PreparerContactName>Jay Cook</PreparerContactName>
        <PreparerTitle>Project Manager</PreparerTitle>
        <PreparerFaxNbr>9725312108</PreparerFaxNbr>
        <PreparerEmailAddress>jcook6@jcpenney.com</PreparerEmailAddress>
    </EmployerTPASeparationResponse>
</EmployerTPASeparationResponseCollection>
```

8.1.1.2.1.1 Duplicate Records

During certification, a test must be run to determine if the connector software can handle duplicate records. Duplicate records can occur for many reasons, and because the Broker consolidates the records it passes on to a connector, the connector software not only has to handle duplicate records in separate SOAP messages, but also within the same SOAP message. It is critical that the connector software be able to handle both cases without fail.

NOTE: Because of the small chance that a GUID will be repeated by different States, employer/TPAs must ensure that both the State Request Record GUID **AND** the State abbreviation (or something similar) be used when determining if a record is a duplicate.

There are data files created to test the connector for its ability to handle duplicates. The state has a Duplicate_Response.xml file, and the employer/TPA has a Duplicate_Request.xml file. These data files must be injected into the connector software to determine if the connector software processes duplicate records appropriately. Since the **SIDES** technical support team cannot validate connector back-end processing, the connector must self-certify its duplicate processing functionality during internal testing. After connector testing, duplicate processing results must be filled into the certification spreadsheet.

8.1.1.2.1.2 Message Codes

As part of the connector certification process, the connector software must be verified that it can handle all message codes returned by the **Central Broker** in the SOAP message header. If the connector software is coded correctly, message codes 2 (example of which is a file containing all invalid records) and message codes 3 (file containing good and invalid records) should not be encountered during certification testing with the **Central Broker** as the data files to be posted are clean. Since the **SIDES** technical support team cannot validate connector back-end processing, the connector must self-certify message codes. After connector testing, results must be filled into the certification spreadsheet.

To support certification of message codes six (6) XML data files were produced. Three files will be used to test requests (state posts) and three files to test responses (employer/TPA posts).

States will use the following files to test the message codes:

- Request_Message_Code_1.xml – file contains good records and returns a message code = 1
- Request_Message_Code_2.xml – file contains 1 bad record, which will return a message code = 2
- Request_Message_Code_3.xml for the States – file contains 1 good record and 1 bad record and it will return a message code = 3

For employers or TPAs:

- Response_Message_Code_1.xml– file contains good records and returns a message code = 1
- Response_Message_Code_2.xml– file contains 1 bad record, which will return a message code = 2
- Response_Message_Code_3.xml – file contains 1 good record and 1 bad record and it will return a message code = 3

8.1.1.2.1.3 Attachments

As part of the connector certification process, the connector must verify that attachments received via the **Central Broker** can be opened successfully and processed by the connector's back-end system (for those exchanges that have attachments; otherwise skip to the next section). Since the **SIDES** technical support team cannot validate connector back-end processing, the connector must self-certify attachment processing. After connector testing, results must be filled into the certification spreadsheet. To support certification of attachments, two (2) XML data files were produced. States will certify attachment processing using the Valid_Response_1.xml file, and employers will use Valid_Request_1.xml file.

8.1.1.2.2 Step 2.2 - Compare Results

After each certification test file is run, the test results should be accessible to the tester. These results can be inside a test log file or stored in the database. The method does not matter to the **SIDES** technical support team as this is connector design specific.

Once the tester has the test results from the connector software, the tester should compare it to the expected results, which is contained in the header of the test file. The results must be an exact match of the expected results. Test files that were created to fail business rules may contain 2 or more errors in them. It is crucial that ALL errors be caught. The connector software or back-end software must also be evaluated to ensure that **Central Broker** error codes or message codes have been handled correctly.

The tester should also determined if the connector software caught different or too many errors. If this is the case, the errors should be examined to determine whether they should have been actually caught. The **SIDES** technical support team should be contacted with the discrepancy if the connector cannot determine the problem (or determines there is a problem in the file).

Example

In the above example, BR210, BR212, BR215, BR217, BR237, BR257, and BR258 should be identified as errors. They must be the only errors identified, but all of them must be identified.

8.1.1.2.3 Step 2.3 - Fill in the Spreadsheet

After the test is completed and the results are satisfactory, the tester should fill in all highlighted cells in both tabs of the worksheet. For the Valid Request / Response Test tab, fill in the cells under the heading "Enter Post Message Code" and "Enter Pull Message Code" with the appropriate message code returned by the **Central Broker.** Since all request / response file pairs are valid requests, success can be determined if all request/response files have a post message code = 1 and a pull message code = 1 and 2 (pulling from the **Central Broker** returns a 1 for the file being returned and a 2 for the empty file)**.**

NOTE: Each request / response file pair must be individually tested (one at a time) before going onto the next request / response file pair. Otherwise, the **Central Broker** will return a single response file that has all responses posted aggregated into a single file.

For the duplicate processing test, fill in the columns “Performed Step? (Yes/No)”, “Did the State System Identify Duplicates? (Yes/No)”, and “How are Duplicates Handled by the Back-End System?”

It is the connector’s responsibility to test the processing of message codes returned by the **Central Broker**. UI agencies, employers, and TPAs must self-certify their message code processing. After running connector in-house testing, fill in the Post Message Code and Pull Message Code tables. For the Post Message Code table, fill in the columns “Did the Connector System Record and Handle the File -Level Message Code Correctly? (Yes/No)” and “Did the Back-End System Update the Individual Records Correctly? (Yes/No)”. Complete the Pull Message Code table by filling in the column “Was the File - Level Pull Message Code handled Correctly? (Yes/No)”.

UI agencies, employers, and TPAs must self-certify their attachment processing (for those exchanges that contain attachments). As part of the preliminary connector certification testing, fill in the Attachment Processing table. Complete the columns “Was the Connector able to Open the Attachment Successfully? (Yes/No)” and “Did the Back-End System Handle the Attachment Correctly? (Yes/No)”. Entering “Yes” in these columns indicate that the attachments were able to be opened successfully and the connector processed the attachments correctly.

For the Invalid Request Test tab, fill in the “Errors detected? (Yes/No)” and the “Back-end Processing Handled Error? (Yes/No)” columns. In the “ Errors detected? (Yes/No)” column, mark “Yes” if the actual output was different than the expected output. Mark “No” if the actual and expected output is the same. The column “Back-end Processing Handled Error (Yes/No)” must also be marked “Yes” if the connector or back-end system did not process the **Central Broker** error code or message code successfully. Mark “No” if it did. Then continue on to the next test.

8.1.1.3 Step 3 - Submit the Spreadsheet

After the tests are complete, the connector should have both tabs of the spreadsheet filled in and the spreadsheet must be submitted to the SIDES Business Manager. Upon receipt of the certification spreadsheet, the SIDES Business Manager and the connector will agree to the date and time period to conduct the final SIDES certification test.

8.1.1.4 Step 4 - Conduct Final Connector Certification Test

When the connector has completed all preliminary certification tests, the final connector certification test is performed.

During the final certification test, the connector will work in conjunction with the SIDES technical support team to coordinate the re-submission of all files in the test suite to the **Central Broker** test environment. The SIDES technical support team will work with the connectors to stage data corresponding to their opposite connector. For example, state connectors will post requests to the **Central Broker** test environment and the SIDES technical support team will emulate the employer and post the matching response files to the **Central Broker** test environment.

The **Central Broker** technical team will perform validation of the certification test through inspection of system logs and examination of file transfer and record detail reports. Certification of duplicate processing, post message codes, and pull message codes will be self-certified by the connector. Self-certification is based on the spreadsheet (provided to the **SIDES** Business Manager), which documents the connector's preliminary connector certification test results.

The final certification test will run over an agreed upon time period. During this certification timeframe, the connector must not send any files to the **Central Broker** otherwise the certification test results will be contaminated with bad data.

Upon successful certification test, the connector will be notified that the software may be promoted into the **SIDES** production environment. If the certification test was not successful, the connector must remediate the problems and re-execute the certification test.

9 COMMON MISTAKES, THINGS TO REMEMBER, KEY DEVELOMENT PITFALLS

9.1 Common Mistakes

9.1.1 Invalid To: and/or From:

One common mistake is to provide the 'TO:' and 'FROM:' SOAP headers in an incorrect manner. The 'TO:' and 'FROM:' must be in the form described in Section 4.3- SOAP Custom Headers.

Most of the instances of this type seen by the **Central Broker** are when the two fields have been reversed.

Also, the 'TO:' on Pulls must be to the Broker and not an individual connector.

9.1.2 Connector Not a Participant

The Unique IDs presented in Section 4.2.1 - Unique ID are the only participants **SIDES** recognizes.

The most common error in this category is using older Unique IDs. (There was a change in Employer/TPA Unique IDs to accommodate the **SIDES** Employer Website.)

Another common error is to omit one or more digits on the employer/TPA Unique ID. It will always be a 'BR' followed by nine digits for a UI **SIDES** Web services participant (as opposed to an UI **SIDES** Employer Website participant – which is the nine-digit Federal Employer Identification Number).

9.1.3 Invalid SOAP Action

Without a properly defined SOAP Action, the **Central Broker** does not know how to process the incoming message. **REDACTED** The state, employer, or TPA connector must handle this situation by its HTTP response timeout-handling routine.

9.1.4 Incorrect/Missing Security

As discussed in Section 5 - C – BUILD THE CONNECTOR: securing the message, security is of paramount concern to the Broker. When first beginning the process to connect with the Broker, the hardest problems encountered involve security. The connector should make sure that it is following all the security preparations for the message. If the connector believes it is doing everything correctly, then it should get in touch with the Broker team to help debug the issue.

9.1.5 Central Broker Not Having Up-to-Date REDACTED Information

REDACTED

9.1.6 Date/Time on Server Not Accurate

The data and time on the server(s) that create the timestamp must be accurate.

REDACTED

9.1.7 Interpretation of Standard Format for Money Fields

Numeric fields defined in a standard format use a fixed-point representation of a number whose total length is represented by the digits to the left of the decimal point and the fractional amount is represented by the digits to the right of the decimal point. For example, the TotalEarnedWages field in Separation Information is defined as numeric 15.2. This represents a number (dollar amount) that can have 13 digits to the left of the decimal point and two digits to the right of the decimal point. The largest number this field can contain is 9,999,999,999,999.99, while the smallest number is 0.00.

9.1.8 State Employer Account Number

As per the SIDES standard formats, UI agencies must pass the state employer account number to employers and TPAs. The StateEmployerAccountNbr is a character field with a maximum length of 20 bytes that the employer or TPA to locate the separating employer in their automated systems. When implementing SIDES, it is essential that the UI agency does not change the StateEmployerAccountNbr that the employers or TPAs presently receives. For example, if an employer or TPA presently receives a StateEmployerAccountNbr without a location code, please ensure that location code is not appended to the StateEmployerAccountNbr. Otherwise, the employer and TPA will not be able to respond to the request for separation information as the employer cannot be looked up on their system.

9.2 Things to Remember

This section discusses some of the issues to be considered by a developer beyond the scope of the connection to the **Central Broker**.

REDACTED

9.2.1 Existing Business System Modifications

Key aspects of a state, employer, or TPA connector solution are having a data store that conforms to the standard separation request and response formats, being able to collect the requisite request data from the claimants to populate the request part of the data store, deliver the standard response data collected from an employer/TPA via the Broker, and integrate imaging/document management technology into the solution.

So, in addition to the communications portion of the connector system, a state connector must be able to consume and generate XML-based data and interact with the state's end users (adjudicators) in terms of delivery of the data to them for processing. For Separation Information., request data may be collected from claimants as part of an Internet Claim Filing application that includes "intelligent" fact finding such that the appropriate question/data flows are presented based on the separation reason chosen by the claimant, or, alternatively, this separation data may be collected during the fact finding by a Call Center agent using screens or Web pages consistent with the separations request format.

It is very likely that employers and TPAs will provide attachments as part of their responses (for exchanges that include attachments). Therefore states will need to be able to consume these attachments, and organize and route them in accordance with their non-monetary adjudication processes.

Finally, as a template for the user interface for delivery of the separation data to state non-monetary adjudication staff for processing, the **SIDES** Employer Website can be an aid, as all the applicable question for each response data element has been defined as well as the page flows for each of the possible 22 separation reasons.

If an agency currently has a paper system you will need to consider what to do with the XML data when it arrives with you. In the worst case, it can be printed out and used in the current manner as in your existing process, but with the benefits of better data quality, no postal delays, and no postage costs.

If an agency has an existing electronic system, a state and employer or TPA will need to make sure that the **SIDES** XML data is interfaced into the target system successfully.

9.2.2 Error Handling

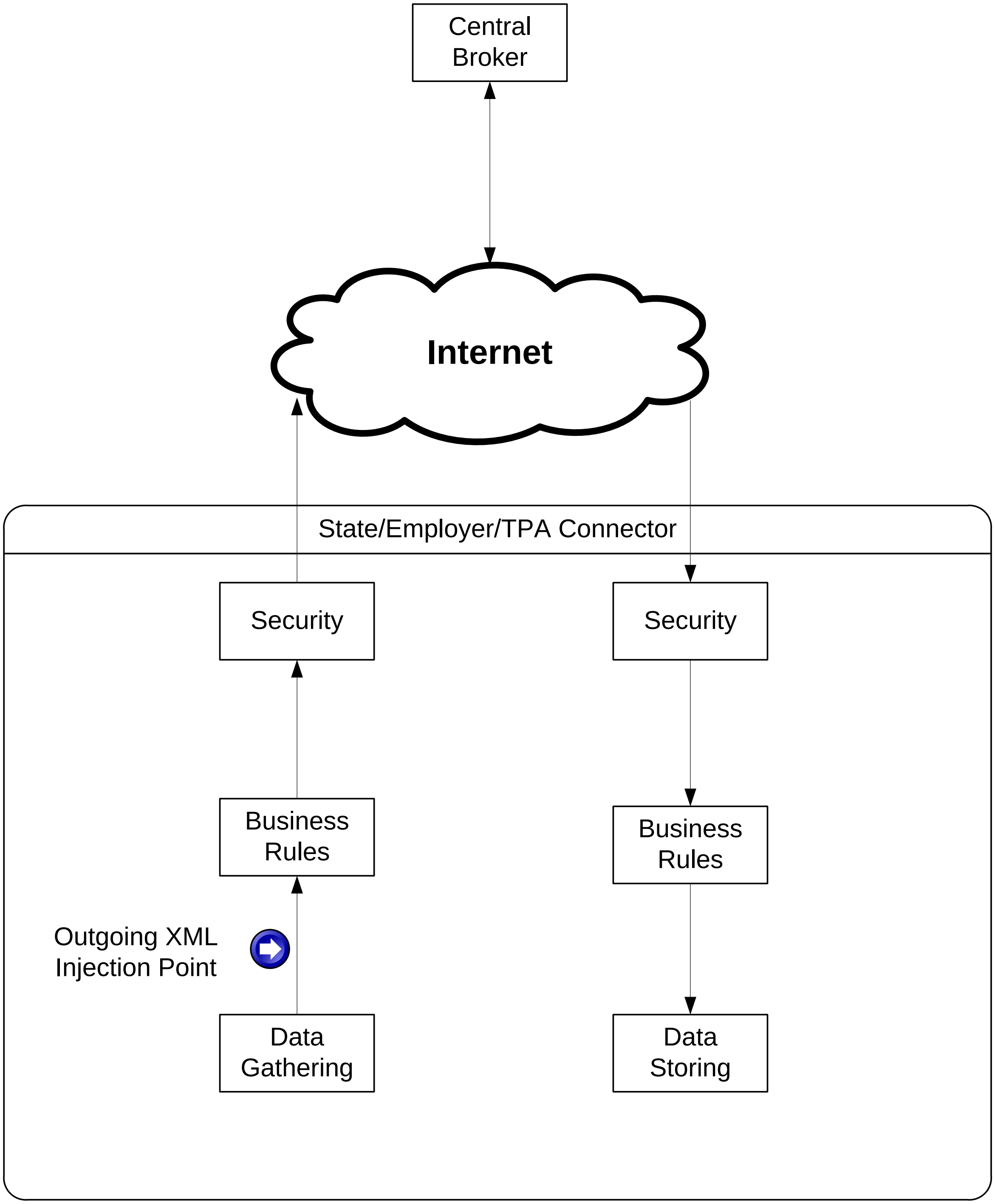
Discovering errors in the data sent to it is one of the important functions performed by the **Central Broker**. But having connectors send records with errors wastes throughput and ties up valuable **Central Broker** resources. This is one of the reasons why it is important that each connector implement the business rules/validation on their systems also.

If, by chance, an error is received from the **Central Broker**, then more than just a correction should be made to that record. A review needs to take place on why that data was allowed to be sent to the **Central Broker** in the first place, and any corrections made to the connectors system must be made (or a global change needs to take place with all connector systems and the **Central Broker** if an inherent flaw is discovered).

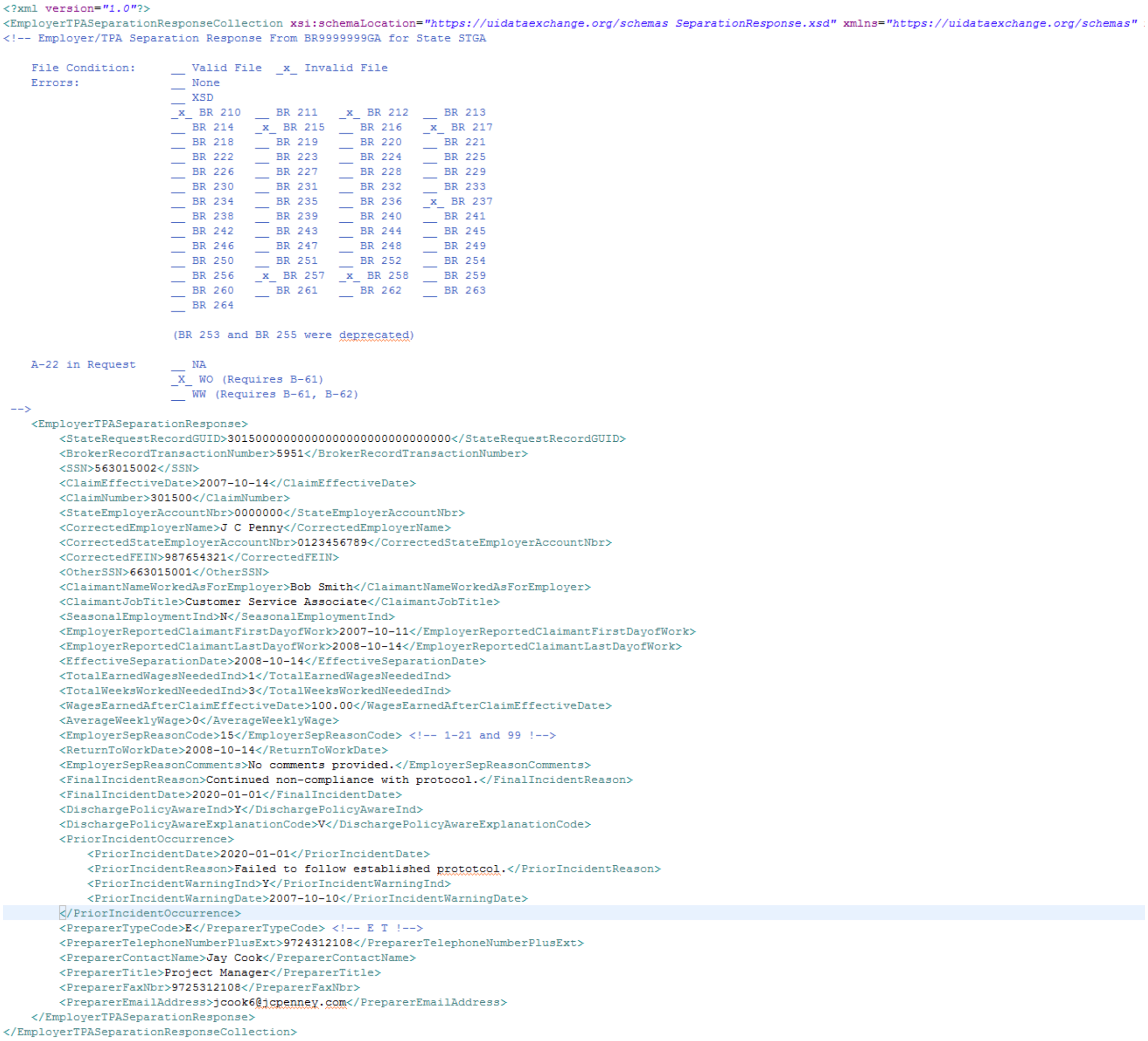
9.2.2.1 XML Injection

To fully test the connector software and its interaction with the **Central Broker**, a set of XML files has been created to simulate possible valid and invalid request and response files than may be generated a member of **SIDES**. For the type of files the connector creates for a post, valid files are to be passed on to the **Central Broker** for processing whereas invalid files are to be caught by the connector software before the messaging takes place with the **Central Broker**. For the type of files the connector accepts on a pull, valid files should be passed to the connectors' backend and invalid files should be caught and flagged.

These XML files make up a certification package that must be run against the connector software to validate that it is ready to run with the **Central Broker** in the production environment. Therefore, these XML files must be able to be injected into the connector software at a point before the business rules are checked within the connector software. States, Employers / TPAs must be able to inject outgoing messages to the **Central Broker**, as the connector software must check all their outgoing business rules.



The XML files given as part of the Certification Process will contain all the information required to determine what, if anything, is wrong with that file. A sample file is included below.



```
<?xml version="1.0"?>
<EmployerTPASeparationResponseCollection xsi:schemaLocation="https://uidataexchange.org/schemas SeparationResponse.xsd" xmlns="https://uidataexchange.org/schemas"
<!-- Employer/TPA Separation Response From BR9999999GA for State STGA

    File Condition:     __ Valid File  _x_ Invalid File
    Errors:             __ None
                        __ XSD
                        _x_ BR 210  __ BR 211   _x_ BR 212  __ BR 213
                        __ BR 214   _x_ BR 215  __ BR 216   _x_ BR 217
                        __ BR 218   __ BR 219   __ BR 220   __ BR 221
                        __ BR 222   __ BR 223   __ BR 224   __ BR 225
                        __ BR 226   __ BR 227   __ BR 228   __ BR 229
                        __ BR 230   __ BR 231   __ BR 232   __ BR 233
                        __ BR 234   __ BR 235   __ BR 236   _x_ BR 237
                        __ BR 238   __ BR 239   __ BR 240   __ BR 241
                        __ BR 242   __ BR 243   __ BR 244   __ BR 245
                        __ BR 246   __ BR 247   __ BR 248   __ BR 249
                        __ BR 250   __ BR 251   __ BR 252   __ BR 254
                        __ BR 256   _x_ BR 257  _x_ BR 258  __ BR 259
                        __ BR 260   __ BR 261   __ BR 262   __ BR 263
                        __ BR 264

                        (BR 253 and BR 255 were deprecated)

    A-22 in Request     __ NA
                        _X_ WO (Requires B-61)
                        __ WW (Requires B-61, B-62)
 -->
    <EmployerTPASeparationResponse>
        <StateRequestRecordGUID>30150000000000000000000000000000</StateRequestRecordGUID>
        <BrokerRecordTransactionNumber>5951</BrokerRecordTransactionNumber>
        <SSN>563015002</SSN>
        <ClaimEffectiveDate>2007-10-14</ClaimEffectiveDate>
        <ClaimNumber>301500</ClaimNumber>
        <StateEmployerAccountNbr>0000000</StateEmployerAccountNbr>
        <CorrectedEmployerName>J C Penny</CorrectedEmployerName>
        <CorrectedStateEmployerAccountNbr>0123456789</CorrectedStateEmployerAccountNbr>
        <CorrectedFEIN>987654321</CorrectedFEIN>
        <OtherSSN>663015001</OtherSSN>
        <ClaimantNameWorkedAsForEmployer>Bob Smith</ClaimantNameWorkedAsForEmployer>
        <ClaimantJobTitle>Customer Service Associate</ClaimantJobTitle>
        <SeasonalEmploymentInd>N</SeasonalEmploymentInd>
        <EmployerReportedClaimantFirstDayofWork>2007-10-11</EmployerReportedClaimantFirstDayofWork>
        <EmployerReportedClaimantLastDayofWork>2008-10-14</EmployerReportedClaimantLastDayofWork>
        <EffectiveSeparationDate>2008-10-14</EffectiveSeparationDate>
        <TotalEarnedWagesNeededInd>1</TotalEarnedWagesNeededInd>
        <TotalWeeksWorkedNeededInd>3</TotalWeeksWorkedNeededInd>
        <WagesEarnedAfterClaimEffectiveDate>100.00</WagesEarnedAfterClaimEffectiveDate>
        <AverageWeeklyWage>0</AverageWeeklyWage>
        <EmployerSepReasonCode>15</EmployerSepReasonCode> <!-- 1-21 and 99 !-->
        <ReturnToWorkDate>2008-10-14</ReturnToWorkDate>
        <EmployerSepReasonComments>No comments provided.</EmployerSepReasonComments>
        <FinalIncidentReason>Continued non-compliance with protocol.</FinalIncidentReason>
        <FinalIncidentDate>2020-01-01</FinalIncidentDate>
        <DischargePolicyAwareInd>Y</DischargePolicyAwareInd>
        <DischargePolicyAwareExplanationCode>V</DischargePolicyAwareExplanationCode>
        <PriorIncidentOccurrence>
            <PriorIncidentDate>2020-01-01</PriorIncidentDate>
            <PriorIncidentReason>Failed to follow established prototcol.</PriorIncidentReason>
            <PriorIncidentWarningInd>Y</PriorIncidentWarningInd>
            <PriorIncidentWarningDate>2007-10-10</PriorIncidentWarningDate>
        </PriorIncidentOccurrence>
        <PreparerTypeCode>E</PreparerTypeCode> <!-- E T !-->
        <PreparerTelephoneNumberPlusExt>9724312108</PreparerTelephoneNumberPlusExt>
        <PreparerContactName>Jay Cook</PreparerContactName>
        <PreparerTitle>Project Manager</PreparerTitle>
        <PreparerFaxNbr>9725312108</PreparerFaxNbr>
        <PreparerEmailAddress>jcook6@jcpenney.com</PreparerEmailAddress>
    </EmployerTPASeparationResponse>
</EmployerTPASeparationResponseCollection>
```

9.2.3 Disaster Recovery

REDACTED

9.3 Key Development Pitfalls

9.3.1 Java – REDACTED

REDACTED

9.3.2 Spring-WS – Timestamp issue

REDACTED

10 LIST OF FIGURES

11 LIST OF TABLES