# Benefit Resolution Analyzer

**Client: Recondo Technology**
**Team: Jared Steele and Tyson Williams**

## 1. Abstract

At Recondo Technology the accuracy process, Recondo’s method of handling any disputed reportings from hospitals and insurance companies, is currently handled by the developers. The company wishes to move this responsibility to the Revenue Cycle Service (RCS) team members. The non-technical RCS team is unable to easily handle this task because much of the process is highly technical and involves using Eclipse and understanding how the Drools rule based expert system works.

Our project involved building a web based graphical user interface (GUI) Benefit Resolution Analyzer service that will allow the RCS team to easily run a transaction against either the deployed rules on a particular customer’s server, or the latest version of rules kept on the Subversion (SVN) repository. The Benefit Resolution Analyzer will then display the rules that the encounter hit, the encounter information, and benefits information; all of which are important identifying information to help in debugging the problem.

To accomplish this we used the Google Web Toolkit (GWT) for the web based GUI. The rest of our project involved connecting to existing databases and repositories to gather the information needed to run a transaction against a user specified set of rules. This was done with SVN, and Java Database Connectivity (JDBC). Along with these two components we also incorporated some interaction with the Drools Rule engine that Recondo uses to run its business rules.

The result of our project is a web application that can be deployed on any Tomcat server. Installation only requires copying the web application archive (WAR) file into the Tomcat server’s webapps folder.

## 2. Introduction

Recondo Technology is a company that provides software as a service for hospitals. This service allows hospitals to give patients an estimate of what their bill will be before they have a procedure, so they can make an educated decision about their personal health care. To do this Recondo handles their business logic using Drools Expert. Drools is a rule engine, a way to translate business rules into code logic, that provides it own language for describing rules. When a transaction is run through the Recondo, system the patient’s insurance information, the procedures they are getting, and other necessary information  is loaded. The rules are run against this data and when finished, the output is the cost of the procedure as well as how much is covered by the patients insurance, their deductible, etc.

When an error occurs, the current accuracy process heavily involves the development team. They are needed to help debug the code, and solve the problems in the rules, as the RCS team doesn't have the ability to decipher code. Recondo would like for the RCS team to handle most of the work; however, the process is currently highly technical and requires using and understanding Eclipse, Subversion, and Drools. Currently Recondo uses a Java class called BenefitResolutionTester to run a transaction and display the output and the rules that were hit. The problem with this method is that you need to manually edit a properties file, and change a few lines of code in the class itself.

The goal for our project was to create a web based GUI that will allow the RCS team to easily analyze a transaction that is not functioning correctly. This GUI would mainly be automating the BenefitResolutionTester, but with more options. For instance, the user could select to run a transaction against either the deployed rules for that customer or the latest rules from SVN. The ability to run the transaction against the deployed rules is quite important, in that it lets the RCS team see exactly what the hospital sees when it ran the transaction.

## 3. Requirements

### A. Functional Requirements

1. Benefit Resolution Analyzer can connect to Operation and IT management (OIT) DB to retrieve connection information for customer DB (maintained by the hospital)
2. Benefit Resolution Analyzer service can connect to any customer DB to retrieve
    a. deployed rules
    b. encounter
    c. benefits
3. Benefit Resolution Analyzer can retrieve latest rules from subversion repository
4. User can upload a modified rule file or a completely new rule file to the application, and run a transaction against the resulting modified rule set
5. The user can run the selected encounter against either the deployed rules on a customer server, or the latest rules from the repository
6. Results are displayed in a meaningful format
    a. encounter
    b. benefits
    c. rules hit
    d. out of pockets and deductibles
7. Rules can be modified by uploading new rule files
8. encounter and benefits can be exported to xml

## B. Non-Functional Requirements

1. All data is stored temporarily and deleted on close
2. Minimize developer interaction with accuracy process
3. Use Maven for project dependency management
4. Compiles as a WAR file to be used on a Tomcat web server
5. Web based GUI to be built with Google Web Toolkit
6. Communication between existing databases and repositories with SVN, JDBC

# 4. Design

## A. High Level Design

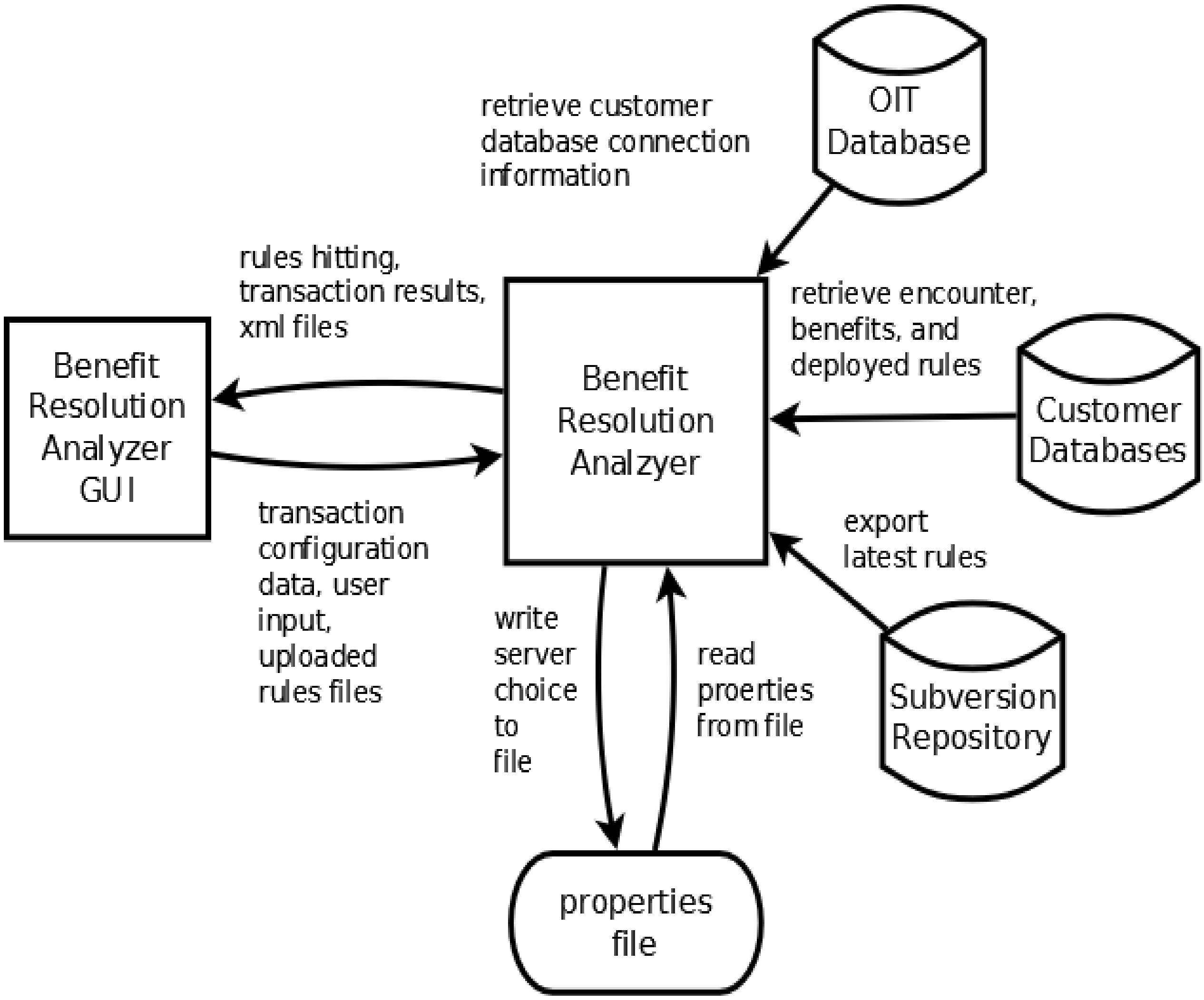
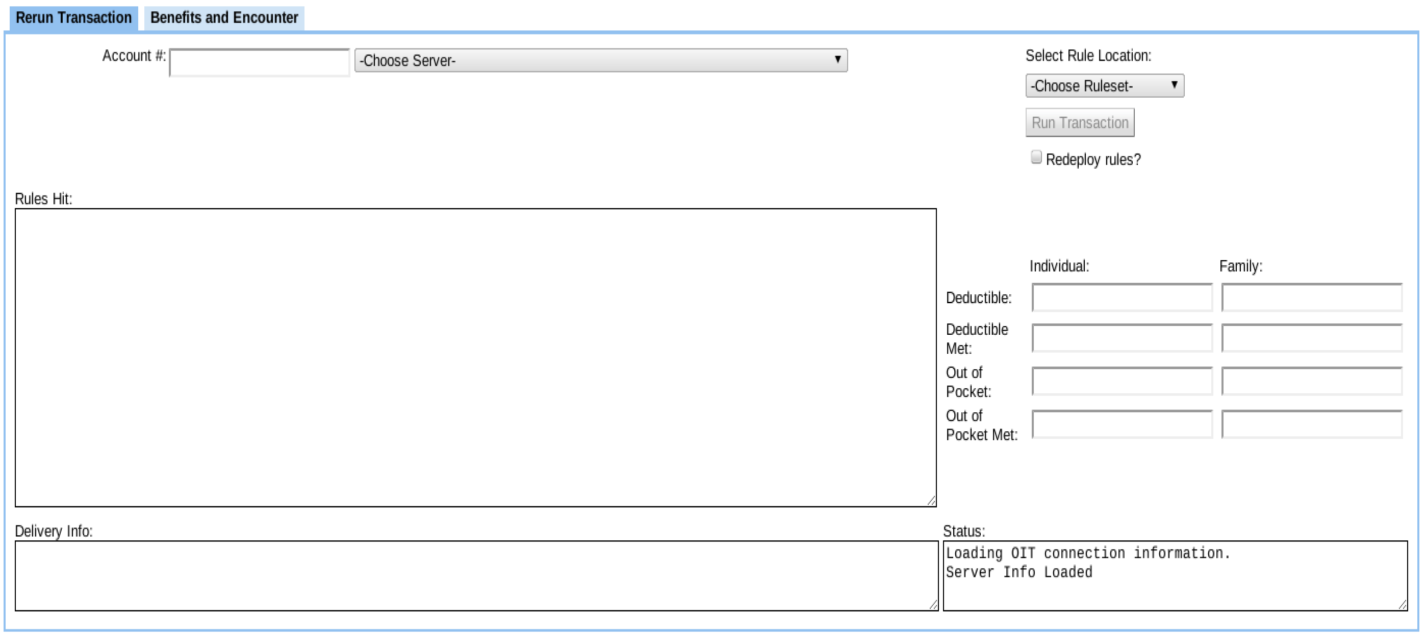

OIT Database
retrieve customer database connection information
rules hitting, transaction results, xml files
Benefit Resolution Analyzer GUI
Benefit Resolution Analzyer
retrieve encounter, benefits, and deployed rules
Customer Databases
transaction configuration data, user input, uploaded rules files
export latest rules
write server choice to file
read proerties from file
Subversion Repository
properties file

Figure 1. System Architecture

The high-level design can be seen above in Figure 1. The Benefit Resolution Analyzer will connect to the OIT database to request connection information for the customer databases. Using this information it will then connect to the user specified customer database to retrieve the encounter, benefits, and other configuration data needed to run the transaction. To do this it must write to a service properties file that holds all of the configuration data needed by the Benefit Resolution Analyzer to access customers servers and correctly identify procedures in internal databases. If the user has selected to use the latest rule set it will be retrieved from the SVN repository, or the user can deployed server rules, which can also be retrieved using the tool. Then the user can run a transaction and see the output in the company standard format; the tool will display the rules that hit in real time. After the transaction the user can also export the encounter and benefits XML files to their local file system.

## B. Detail Design

### Benefit Resolution Analyzer GUI



```
Loading OIT connection information.
Server Info Loaded
```

Figure 2. A screenshot of the main page of the GUI

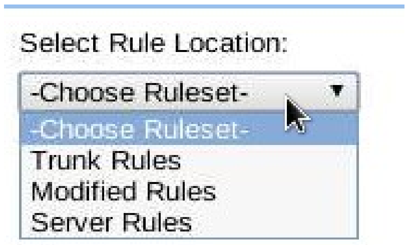


Figure 3. A screenshot of the Rule Selection Dropbox

Figure 2 above shows the basic Graphical User Interface that we designed to be easy to use for the user. There are two basic parts to the GUI: the top which is where the user inputs the information and the bottom where information is provided to the user from the service. The top has two main parts, the first being the area for account number entry and server selection, which will be the identifying information for the transaction. The other part is the selection of the rules, which allows the user to choose which set of rules the transaction should be run against, as show in figure 3 above. There is also a check box that asks the user if they want to redeploy the rules or if the rules should be used from the last run.

The bottom area is broken into 4 parts. The first is the large Text area where the GUI displays the rules that have been hit. To the right of this is the area where the out of pocket and deductible for the specific encounter is displayed. Below the ‘Rules Hit’ text box, is the ‘Delivery Info’ text box which displays relevant information about the encounter based on the needs for Recondo to look up any information in their systems. Finally in the bottom right corner is the ‘Status’ text box, which displays the logging information which tells the user exactly what is being done at the moment by the Benefit Resolution Analyzer.

**OITConnector**
In order to gather the necessary connection information for Recondo’s customer databases we need a class to handle the connection to the OIT database. The OITConnector class sets up a JDBC connection to the OIT database then runs a query to retrieve all the rows needed from the database. The important fields are database name (usually related to the hospital name), tenant ID (a quick identifying number), nickname (the hospital name), realm (whether it’s the production server or not), and url. The results from the query are put into a Row class.

**Row**
The Row class was created to easily handle and deliver formatted connection info to the ServerPropertiesWriter and the BenefitResolutionTester. It has methods to deliver the necessary data to each of these classes.

**ServerPropertiesWriter**
When a user selects the customer’s server to run a transaction with, this selection must be written to a service.properties file. This is done by taking a template file called service.properties_template, replacing a place holder line from with the server data, and writing to the service.properties file, leaving the template unchanged.

**BenefitResolutionTester**
The BenefitResolutionTester is the most important class in the Benefit Resolution Analyzer because it handles deploying the rules, finding the encounter and benefits information, and running this data through the rules. Most of the class is similar to what was in the original class in the Eligibility Service. One change was that we added several different ArrayList<String> to return information from the

BenefitResolutionTester back to our GUI to give the user information on the current status of the transaction. Another change was to add the ability to deploy the rules from a customer’s server and not local files. We also removed much of the Logging system that Recondo was using and replaced it with our own method of logging. Lastly, we added the ability for it to run a modified set of local rules.

**SVNConnector**

The SVN connector handles retrieving the trunk rules from the Recondo SVN repository. We used the SVNKit to establish a connection to the repository and then export all the rule files to the root directory of the Benefit Resolution Analyzer. We used export because, unlike checkout, it does not have any hidden .svn files in the directory that can cause problems with deploying the rules. Also before the export is run any previous rule files in the target directory are deleted to avoid any conflicts. This is to prevent modified files from interfering with running the trunk rules. The only thing needed by the SVNConnector to run is a username and password, which are input by the user to the SVNLoginBox.

**SVNLoginBox**

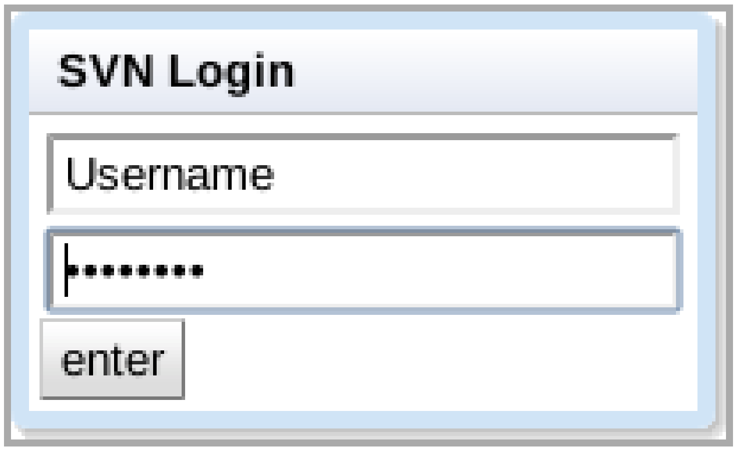


Figure 4. The SVNLoginBox GUI element

As shown in Figure 4, the SVNLoginBox is a DialogBox with a text and password field for the user’s SVN credentials. After the credentials are entered by the user, the SVNLoginBox uses them to make the Remote Procedure Call (RPC) to the SVNConnector to retrieve the rule files.

**FileUploadServlet**

The FileUploadServlet allows for the uploading of modified rule files or completely new rule files. This connects to the ‘Select Rule Location’ drop down list on the GUI shown in figure 3. More specifically, it reads the HTML form generated by the FileUploadBox and saves the uploaded file to the correct scope in the rules directory. This is accomplished with the help of the Apache commons-fileupload library.

**FileUploadBox**

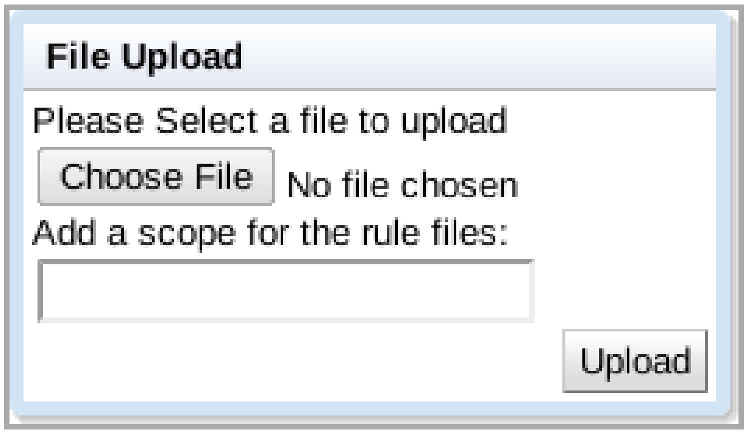

File Upload
Please Select a file to upload
Choose File
No file chosen
Add a scope for the rule files:
Upload

Figure 5. The FileUploadBox GUI element

As shown in Figure 5, the FileUploadBox is similar to the SVNLoginBox because it is also a DialogBox. The Choose File button on it opens a file browser where you can select a file to upload. The scope box is required so that the FileUploadServlet save the file into the right scope in the rules directory. Otherwise the rules can not be deployed correctly.

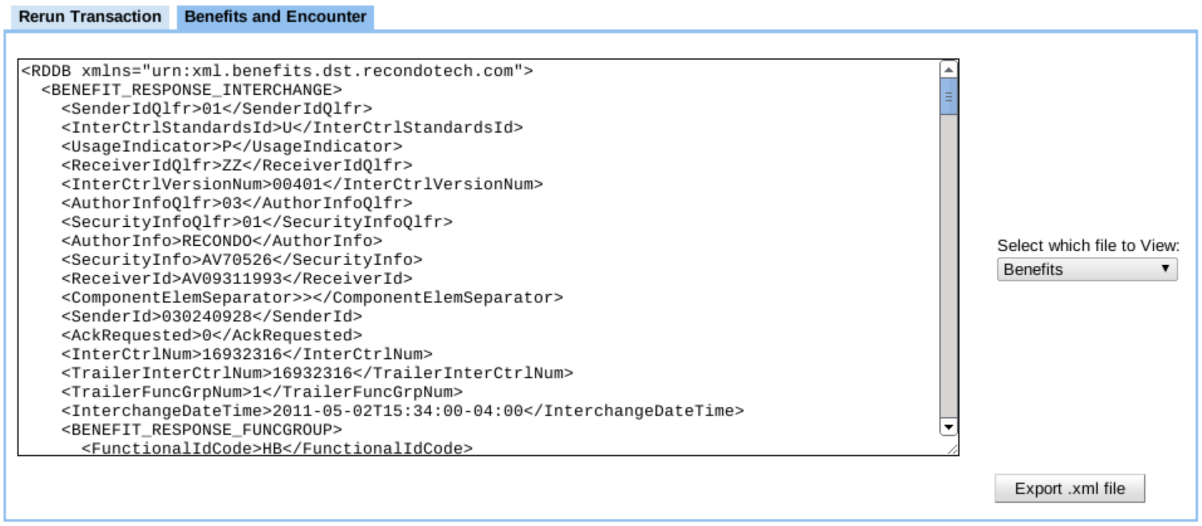
**Global**

The Global class was created to hold the webapp_root() function that returns the system dependent path to the Tomcat base directory. This allows the webapp to be deployed on Windows or Linux without any errors. Global also holds a recursive delete method that deletes the rules on close or before the SVN export is done.

**RuleService, RuleServiceAsync, RuleServiceImpl**

These are the three classes required by GWT for the RPC to work. The contain all the RPC methods we used, and the implementation of these methods is found in the RuleServiceImpl file in the server side code.

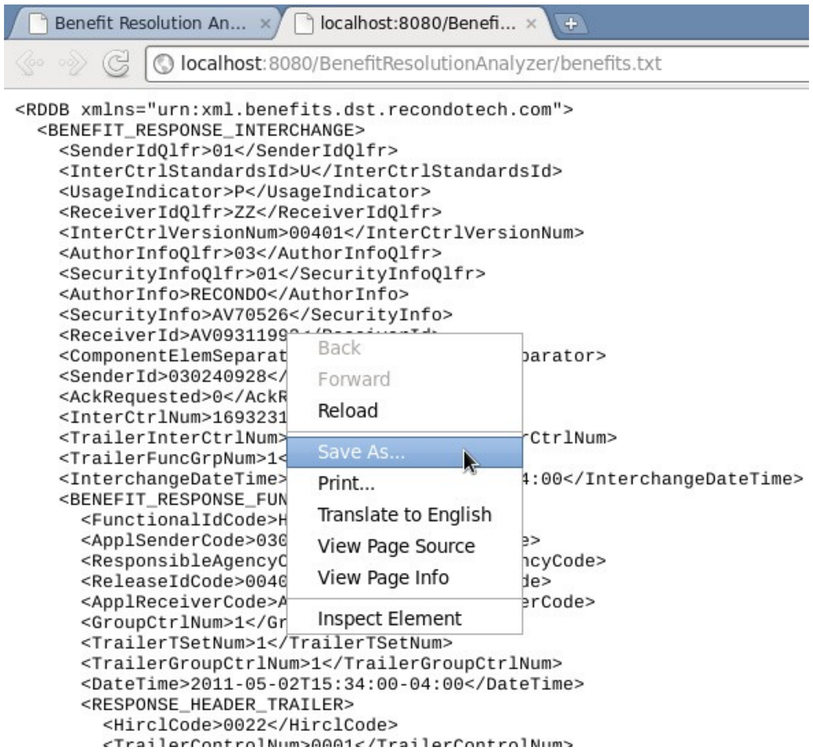
## Viewing and Exporting XML



```
<RDDB xmlns="urn:xml.benefits.dst.recondotech.com">
  <BENEFIT_RESPONSE_INTERCHANGE>
    <SenderIdQlfr>01</SenderIdQlfr>
    <InterCtrlStandardsId>U</InterCtrlStandardsId>
    <UsageIndicator>P</UsageIndicator>
    <ReceiverIdQlfr>ZZ</ReceiverIdQlfr>
    <InterCtrlVersionNum>00401</InterCtrlVersionNum>
    <AuthorInfoQlfr>03</AuthorInfoQlfr>
    <SecurityInfoQlfr>01</SecurityInfoQlfr>
    <AuthorInfo>RECONDO</AuthorInfo>
    <SecurityInfo>AV70526</SecurityInfo>
    <ReceiverId>AV09311993</ReceiverId>
    <ComponentElemSeparator>></ComponentElemSeparator>
    <SenderId>030240928</SenderId>
    <AckRequested>0</AckRequested>
    <InterCtrlNum>16932316</InterCtrlNum>
    <TrailerInterCtrlNum>16932316</TrailerInterCtrlNum>
    <TrailerFuncGrpNum>1</TrailerFuncGrpNum>
    <InterchangeDateTime>2011-05-02T15:34:00-04:00</InterchangeDateTime>
    <BENEFIT_RESPONSE_FUNCGROUP>
      <FunctionalIdCode>HB</FunctionalIdCode>
```

Figure 6. Benefits and Encounter Tab

While not in a class of its own, this important feature is located on the second tab at the top of the GUI, as shown in Figure 6. In the BenefitResolutionTester, the XML data for the benefits and encounter information is written to a file in the base directory of the webapp. From the file selection box on the Benefits and Encounter tab you can view the files in the text box as shown in Figure 6, or when export is clicked a savable copy will open in the user’s browser for them to save locally, as shown below in Figure 7. The user just needs to right click and select ‘Save As’ to open a file browser window.



```
<RDDB xmlns="urn:xml.benefits.dst.recondotech.com">
  <BENEFIT_RESPONSE_INTERCHANGE>
    <SenderIdQlfr>01</SenderIdQlfr>
    <InterCtrlStandardsId>U</InterCtrlStandardsId>
    <UsageIndicator>P</UsageIndicator>
    <ReceiverIdQlfr>ZZ</ReceiverIdQlfr>
    <InterCtrlVersionNum>00401</InterCtrlVersionNum>
    <AuthorInfoQlfr>03</AuthorInfoQlfr>
    <SecurityInfoQlfr>01</SecurityInfoQlfr>
    <AuthorInfo>RECONDO</AuthorInfo>
    <SecurityInfo>AV70526</SecurityInfo>
    <ReceiverId>AV09311993
    <ComponentElemSeparat                         parator>
    <SenderId>030240928</
    <AckRequested>0</AckR
    <InterCtrlNum>1693231
    <TrailerInterCtrlNum>                         rCtrlNum>
    <TrailerFuncGrpNum>1<
    <InterchangeDateTime>                         4:00</InterchangeDateTime>
    <BENEFIT_RESPONSE_FUN
      <FunctionalIdCode>H
      <ApplSenderCode>030                         e>
      <ResponsibleAgencyC                         ncyCode>
      <ReleaseIdCode>0040                         de>
      <ApplReceiverCode>A                         erCode>
      <GroupCtrlNum>1</Gr
      <TrailerTSetNum>1</TrailerTSetNum>
      <TrailerGroupCtrlNum>1</TrailerGroupCtrlNum>
      <DateTime>2011-05-02T15:34:00-04:00</DateTime>
      <RESPONSE_HEADER_TRAILER>
        <HirclCode>0022</HirclCode>
        <TrailerControlNum>0001</TrailerControlNum>
```

Figure 7. Export XML screenshot

## 5. Project Progression and Scope

### A. Successful implementation

Recondo Technologies needed a way to test their transactions to ensure their accuracy. This was the main scope and goal of the project. The testing of transactions had several parts that needed to be handled. The first of these was to automate the server selection process, because Recondo has several hundred companies that it works with, and each of these have at least one server where the transaction information is kept. After the server selection information is retrieved, the next step is selecting the set of rules that the transaction would be run against. The options they requested us to have for the rule sets were the rules that are considered the “gold standard” (the rules on their SVN repository), a modified set of rules that is based on the gold standard rules that the user uploads, or the rules currently on the selected server. Each of these posed their own issue that had to be addressed, such as how they were to be deployed and managed. From here we needed a method to listen to the rule manager that would be running the rules. There were some listeners that were already available in the Drools rule engine, so we managed to just use one of them with some slight modifications. After that, there was just some communication between the service on the server side and the GUI on the client side that had to be dealt with. Using the built in RPC of GWT, this implementation wasn’t too difficult. Along with the RPC, a specific timer had to be implemented that allowed the communication between the two pieces to be both real time and easily displayed to the user.

### B. Determined out of scope

There was really only one major feature that Recondo Technologies wanted from this project that we were unable to deliver on. They had requested that multiple users be able to run the program when it was deployed to their servers. We were unable to deliver on this request specifically because of their internal code structure. The extensive use of static objects and methods caused interference between the users as these methods and objects would overlap and cause the service to return incorrect results. This one issue however is not currently a major issue for the company as the team that will be running the tool is quite small and would probably only need one instance of the tool running at once. Some tweaking of their internal code structure would allow the tool to be run by multiple users and would help fix some other internal issues that could arise from these static methods.

## 6. Future Direction

Recondo has already made big plans for this project, hoping to help further streamline their accuracy process. They have another tool that a student interning from Mines has written that they want to incorporate into the Benefit Resolution Analyzer to give it more functionality. The tool allows Recondo to watch all the information and transactions through a given running service. This would allow Recondo to log many transactions and look at the logs if there were issues. It also allows for comparing rule files to see the differences between them, allowing the user to know what is different between a rule file that fixed a problem and one that did not. This, incorporated with our existing project, would make a much faster and efficient system for solving problems. Another feature Recondo is looking to add is to allow the user to run their modified rules with the FinishedRulesTester class that holds all their unit tests for the rules. This would allow the user to know if their modification fails any tests.

## 7. Glossary

BRA: Benefit Resolution Analyzer, the name of the tool that was created by the project.

Drools: A functional rules language built in Java to help integrate business rules into programming terms

Encounter: The information relating to the patient that Recondo uses for identification and testing, it helps the RCS team debug problems.

GWT: Google web toolkit, a web protocol to allow web based applications to be easily devolved in Java.

JDBC: Java Data Base Connectivity, a way of querying databases to retrieve information with Java.

Maven: software project management and comprehension tool, Helps with dependency management.

OIT: Operations and IT management system, the databases where the information for the other servers are stored.

RCS: Revenue Cycle Service, Recondo’s testing and customer service group that will be using the Benefit Resolution Analyzer.

RPC: Remote Procedure Call, GWT’s way of communication between the client and server that uses the AsyncCallBack to communicate.

SVN: Subversion, a collaboration tool that allows code to be updated and shared easily

Tomcat**:** an open source software implementation of the Java Servlet and JavaServer Pages technologies, used to host our application.

WAR: Web Application Archive, an archive file for compressing web applications, unpacks into a web application