

|  |
| --- |
| **Tool/Framework – Module Name**  Beneficiary’s Name, Designation – Domain  March 2014 |
|  |

Table of Contents

[Introduction 3](#_Toc478485824)

[Document Purpose 3](#_Toc478485825)

[Current State 3](#_Toc478485826)

[Future State 4](#_Toc478485827)

[Development Environment Setup 5](#_Toc478485828)

[ Configuration: BusinessRule.cfg.xml File 5](#_Toc478485829)

[ Configuration: TOOL/FRAMEWORK.xmlFile 6](#_Toc478485830)

[Code Changes 6](#_Toc478485831)

[ Rule Engine Wrapper Service 6](#_Toc478485832)

[ New Business Rule DB SP 7](#_Toc478485833)

[ DAO (Implementation) Class 9](#_Toc478485834)

[ DAO (Implementation) Class Using Wrapper 11](#_Toc478485835)

[Enhancement Impact 12](#_Toc478485836)

[ Conclusion 13](#_Toc478485837)

# Introduction

The Rule Engine of the TOOL/FRAMEWORK tool is the core unite of this framework that internally deals with voice business profiles and provides extensive test coverage. All the voice business rules and profiles are pre-defined in this module; however, some specific new business rules are to be enhanced and implemented with extra mechanisms (Rule Engine Wrapper Service) around the Rule Engine and Execution Unit.

# Document Purpose

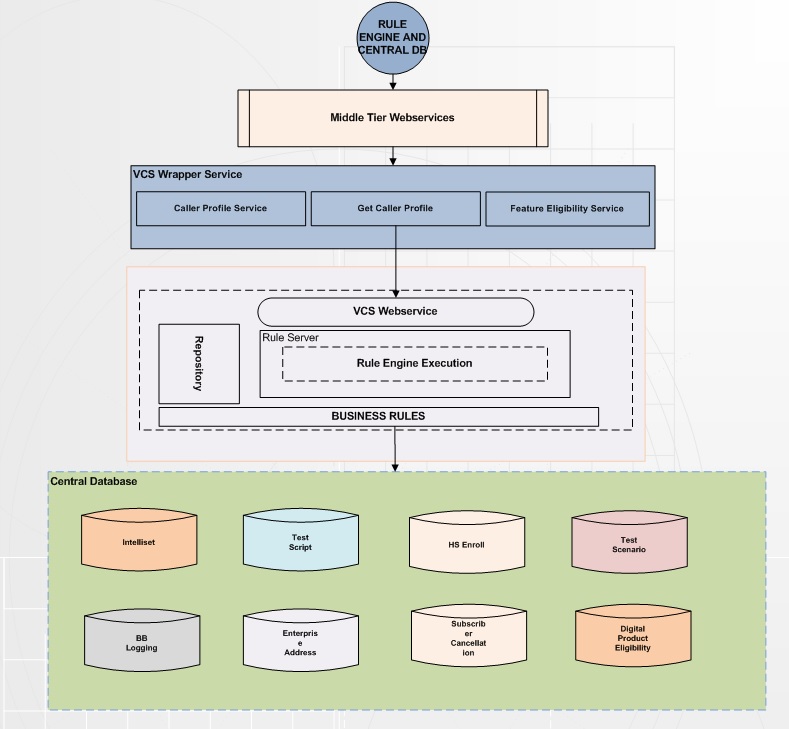
The purpose of this document is to introduce Rule Engine Wrapper Service around the Rule Engine and Execution Unit. The document provides the details of the enhancements and implementation techniques analyzed for the TOOL/FRAMEWORK Rule Engine Module. The document also provides the impact analysis, details of code and configuration and DB Stored Procedure changes, etc.

Rule Engine needs to be updated to accommodate below new business rules. This will include 8 business rules to be added in the TOOL/FRAMEWORK Rule Engine. This will require to implement Rule Engine Wrapper Service to update Rule Engine Repository.

|  |  |  |
| --- | --- | --- |
| **New Business Rule** | | |
| **S. No** | **Rule Id** | **Business Rule Description** |
| 1 | 77964 | Miscellaneous Validation |
| 2 | 77965 | Void-LossDatePurchase |
| 3 | 77967 | VerifyCoverage |
| 4 | 77969 | ClaimLimit |
| 5 | 77971 | Disclaim-PreExistingDamage |
| 6 | 77973 | Shipping Address Validation |
| 7 | 77975 | Document Request |
| 8 | 77977 | ClaimValidation |

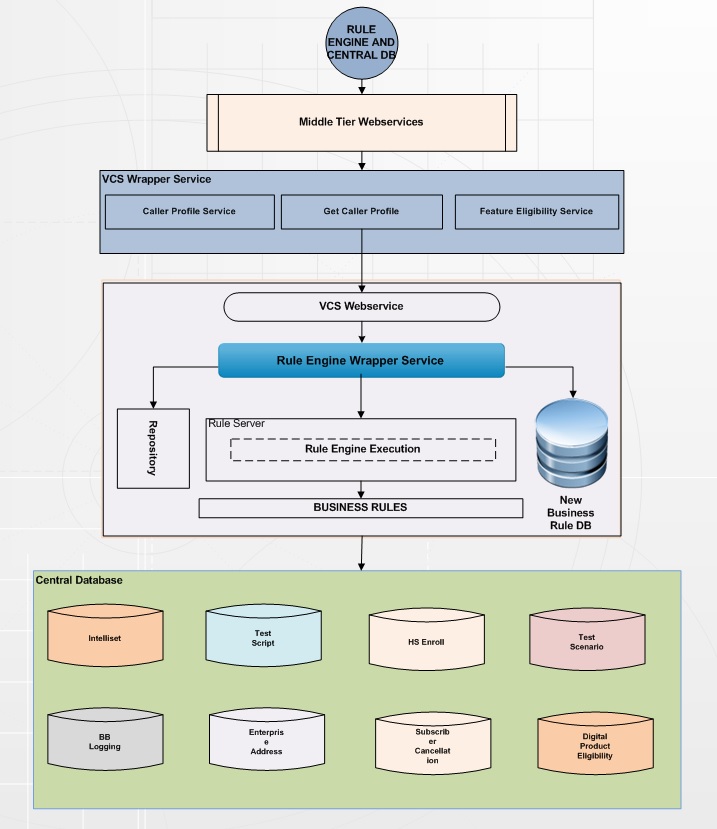
# Current State

As per current enhancement and implementation, Business Rules are getting updated from Central Database (Please refer below current architecture of Rule Engine).



# Future State

The proposed enhancement in the Rule Engine module needs to be updated to accommodate new business rule. To perform this enhancement, a new upgrade mechanism will be designed called Rule Engine Wrapper Service to update and implement Repository that will help… (Please refer below architecture designed for Rule Engine Enhancement).



# Development Environment Setup

This section provides the details for setting up development environment to update Rule Engine module Repository through Rule Engine Wrapper Service upgrade mechanism.

The following configurations are required to be implemented to enhance and upgrade the Rule Engine module:

### Configuration: BusinessRule.cfg.xml File

Context Configurations will contain TOOL/FRAMEWORK context configurations.

<ContextConfigurations>

<Culture>en-US</Culture>

<SystemUsername>TOOL/FRAMEWORKAdmin</SystemUsername>

<SystemPassword>XXXXX</SystemPassword>

<Application>TOOL/FRAMEWORK</Application>

<Channel>Voice</Channel>

</ContextConfigurations>

### Configuration: TOOL/FRAMEWORK.xmlFile

The TOOL/FRAMEWORK.xml files needs to be created and configured as follows –

DefaultConfigurations are declared in this section and values of these configurations could be overridden from the values declared in Services nodes.

<DefaultConfigurations>

<!--Time in ms-->

<ResponseTimeout>40000</TIBCOResponseTimeout>

<MaskingElements>FIRSTNAME,LASTNAME</MaskingElements>

<PaymentApplication>WITSv4</PaymentApplication>

<!-- NonPersistent = 1, Persistent = 2, ReliableDelivery = 22 -->

<QueueDeliveryMode>2</QueueDeliveryMode>

<IsSSL>true</IsSSL>

<IsAsyncCall>False</IsAsyncCall >

</DefaultConfigurations>

# Code Changes

### Rule Engine Wrapper Service

Rule Engine Wrapper Service will fetch all the new business rules DB and update Rule Engine repository. In the current implementation, the domain model classes are simple POJO classes without any framework-specific annotations. Following is one of the sample domain model classes –

**Publicclass** Product {

**privateint**userId;

**private** String firstName;

**private** String lastName;

**private** String email;

**publicint** getUserId() {

**return**userId;

}

**publicvoid** setUserId(**int**userId) {

**this**.userId = userId;

}

**public** String getFirstName() {

**return**firstName;

}

**publicvoid** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return**lastName;

}

**publicvoid** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** String getEmail() {

**return**email;

}

**publicvoid** setEmail(String email) {

**this**.email = email;

}

**publicvoid** setClient(String client) {

**this**.client = alleast;

}

**publicvoid** setRule(String rule) {

**this**.rule = eastrule;

}

@Override

**public** String toString() {

**return**"User [userId=" + userId + ", firstName=" + firstName

+ ", lastName=" + lastName + ", email=" + email + ",

+client "+eastrule"]";

}

**publicvoid** ConfigureRuleToDBd(String rule) {

**this**.rule = this.eastrule;

}

}

### New Business Rule DB SP

<StoredProcedure Name="SET\_NEW\_EAST\_RULE">

<SPName>eastnewrule</SPName>

<Connection>EastProductRule</Connection>

<DBExecuteType>ExecuteScalar</DBExecuteType>

<InputParameters>

<DataClient>@EntryClient</ DataClient>

<PoolGroup>@PoolGroup</PoolGroup>

<DataValue>@DataValue</DataValue>

</InputParameters>

<RequestXSLTRequired>true</RequestXSLTRequired> <RequestXSLTTemplate>EAST\IVR\XSLT\DBServiceRequest\SetInteractionData.xslt</RequestXSLTTemplate> <ScalarOutput>Rules</ScalarOutput>

</StoredProcedure>

============================================================================

<StoredProcedure Name="CLEAR\_RULE\_CACHE">

<SPName>dServiceCache\_byServiceParameter</SPName>

<Connection> EastProductRule</Connection>

<DBExecuteType>ExecuteNonQuery</DBExecuteType>

<InputParameters>

<client\_id>@client\_id</client\_id>

<ServiceParameter>@ServiceParameter</ServiceParameter>

</InputParameters>

<RequestXSLTRequired>true</RequestXSLTRequired> <RequestXSLTTemplate>East\IVR\XSLT\DBServiceRequest\ClearProfileCache.xslt</RequestXSLTTemplate> </StoredProcedure>

The following table contains the description of the annotations used in the above example –

|  |  |  |
| --- | --- | --- |
| **#** | **Annotation** | **Description** |
| 1 | @Entity | Indicates to the persistence of the Tool/Framework framework that the User is indeed a persistent class whose state can be managed by Rule Engine Wrapper Service |
| 2 | @Table | Specifies, to which database table the object maps to. In the above example, the User object maps to the ‘XYZ\_USER’ table of the database |
| 3 | @ServiceParameter | Provides specification of generation strategies for the values of primary keys.  The ServiceParameter annotation may be applied to a primary key property or field of an entity or mapped superclass in conjunction with the Id annotation. The use of the ServiceParameter annotation is only required to be supported for simple primary keys. Use of the ServiceParameter annotation is not supported for derived primary keys. |
| 4 | @Client\_id | Specifies the mapped column for a persistent property or field. If no Client\_id annotation is specified, the default values apply. |

### DAO (Implementation) Class

Following is one of the Data Access Object (DAO) Implementation classes as per the implementation –

**publicclass**UserDAOImpl**implements**UserDAO {

**private**Connectionconn;

**public** UserDAOImpl() {

conn = DBUtil.getConnection();

}

@Override

**publicvoid** addUser( Useruser ) {

**try** {

String query = "insert into XYZ\_USER (firstName, lastName, email) values (?,?,?,?)";

PreparedStatementpreparedStatement = conn.prepareStatement( query );

preparedStatement.setString( 1, user.getFirstName() );

preparedStatement.setString( 2, user.getLastName() );

preparedStatement.setString( 3, user.getEmail() );

preparedStatement.setString( 4, user.getClient() );

preparedStatement.setString( 5, user.getRule() );

preparedStatement.executeUpdate();

preparedStatement.close();

} **catch** (SQLExceptione) {

e.printStackTrace();

}

}

@Override

**publicvoid**deleteUser( **int**userId ) {

**try** {

String query = "delete from EML\_USER where userId=?";

PreparedStatementpreparedStatement = conn.prepareStatement(query);

preparedStatement.setInt(1, userId);

preparedStatement.executeUpdate();

preparedStatement.close();

} **catch** (SQLExceptione) {

e.printStackTrace();

}

}

@Override

**publicvoid** updateUser( Useruser ) {

**try** {

String query = "update XYZ\_USER set firstName=?, lastName=?, email=? where userId=?";

PreparedStatementpreparedStatement = conn.prepareStatement( query );

preparedStatement.setString( 1, user.getFirstName() );

preparedStatement.setString( 2, user.getLastName() );

preparedStatement.setString( 3, user.getEmail() );

preparedStatement.setInt(5, user.getUserId());

preparedStatement.setString( 4, user.getClient() );

preparedStatement.setString( 5, user.getRule() );

preparedStatement.executeUpdate();

preparedStatement.close();

} **catch** (SQLExceptione) {

e.printStackTrace();

}

}

@Override

**public**List<User>getAllUsers() {

List<User>users = **new**ArrayList<User>();

**try** {

Statementstatement = conn.createStatement();

ResultSetresultSet = statement.executeQuery( "select \* from XYZ\_USER" );

**while**( resultSet.next() ) {

Useruser = **new**User();

user.setUserId( resultSet.getInt( "userId" ) );

user.setFirstName( resultSet.getString( "firstName" ) );

user.setLastName( resultSet.getString( "lastName" ) );

user.setEmail( resultSet.getString( "email" ) );

user.setClient( resultSet.getString( "client" ) );

user.setRule( resultSet.getInt( "rule" ) );

users.add(user);

}

resultSet.close();

statement.close();

} **catch** (SQLExceptione) {

e.printStackTrace();

}

**return**users;

}

@Override

**public**UsergetUserById(**int**userId) {

Useruser = **new**User();

**try** {

String query = "select \* from XYZ\_USER where userId=?";

PreparedStatementpreparedStatement = conn.prepareStatement( query );

preparedStatement.setInt(1, userId);

ResultSetresultSet = preparedStatement.executeQuery();

**while**( resultSet.next() ) {

user.setUserId( resultSet.getInt( "userId" ) );

user.setFirstName( resultSet.getString( "firstName" ) );

user.setLastName( resultSet.getString( "LastName" ) );

user.setEmail( resultSet.getString( "email" ) );

user.setClient( resultSet.getString( "client" ) );

user.setRule( resultSet.getInt( "rule" ) );

}

resultSet.close();

preparedStatement.close();

} **catch** (SQLExceptione) {

e.printStackTrace();

}

**return**user;

}

}

### DAO (Implementation) Class Using Wrapper

**publicclass**UserDAOImpl**implements**UserDAO {

**publicvoid** addUser(Useruser) {

Transactiontrns = **null**;

Sessionsession = RuleEngineWrapperUtil.getSessionFactory().openSession();

**try** {

trns = session.beginTransaction();

session.save(user);

session.getTransaction().commit();

} **catch** (RuntimeException e) {

**if** (trns != **null**) {

trns.rollback();

}

e.printStackTrace();

} **finally** {

session.flush();

session.close();

}

}

**publicvoid** deleteUser(**int**userid) {

Transactiontrns = **null**;

Sessionsession = RuleEngineWrapperUtil.getSessionFactory().openSession();

**try** {

trns = session.beginTransaction();

Useruser = (User) session.load(User.**class**, **new** Integer(userid));

session.delete(user);

session.getTransaction().commit();

} **catch** (RuntimeException e) {

**if** (trns != **null**) {

trns.rollback();

}

e.printStackTrace();

} **finally** {

session.flush();

session.close();

}

}

**publicvoid** updateUser(Useruser) {

Transactiontrns = **null**;

Sessionsession = RuleEngineWrapperUtil.getSessionFactory().openSession();

**try** {

trns = session.beginTransaction();

session.update(user);

session.getTransaction().commit();

} **catch** (RuntimeException e) {

**if** (trns != **null**) {

trns.rollback();

}

e.printStackTrace();

} **finally** {

session.flush();

session.close();

}

}

**public**List<User> getAllUsers() {

List<User>users = **new**ArrayList<User>();

Transactiontrns = **null**;

Sessionsession = RuleEngineWrapperUtil.getSessionFactory().openSession();

**try** {

trns = session.beginTransaction();

users = session.createQuery("from User").list();

} **catch** (RuntimeException e) {

e.printStackTrace();

} **finally** {

session.flush();

session.close();

}

**return**users;

}

**public**User getUserById(**int**userid) {

Useruser = **null**;

Transactiontrns = **null**;

Sessionsession = RuleEngineWrapperUtil.getSessionFactory().openSession();

**try** {

trns = session.beginTransaction();

String queryString = "from User where id = :id";

Queryquery = session.createQuery(queryString);

query.setInteger("id", userid);

user = (User) query.uniqueResult();

} **catch** (RuntimeException e) {

e.printStackTrace();

} **finally** {

session.flush();

session.close();

}

**return**user;

}

}

# Enhancement Impact

As mentioned earlier, Rule Engine Wrapper Service is solution mainly used for accommodating new Business rule. Following are the main advantages of a Wrapper Service Implementation.

* **More Productive:** Rule Engine Wrapper Service will help to implement upcoming new business rules very quickly, by implementing this wrapper in the Tool/Framework framework, this will be more productive, robust, and trusted and very loosely decoupled from any other interface.
* **Well Designed:** Rule Engine is an effective architectural design pattern; it is implemented and well tested over a period. We need not to spend our time in performing redundant efforts.

### Conclusion

Based on the above analysis, we can conclude that, implementing the Rule Engine Wrapper Service of the Tool/Framework framework will help in accommodating new business rule changes into the database easily and quickly. It will also improve the code quality and will result into reduced maintenance cost of the Tool/Framework framework. We can also leverage the caching feature to improve the performance of the Tool/Framework framework.