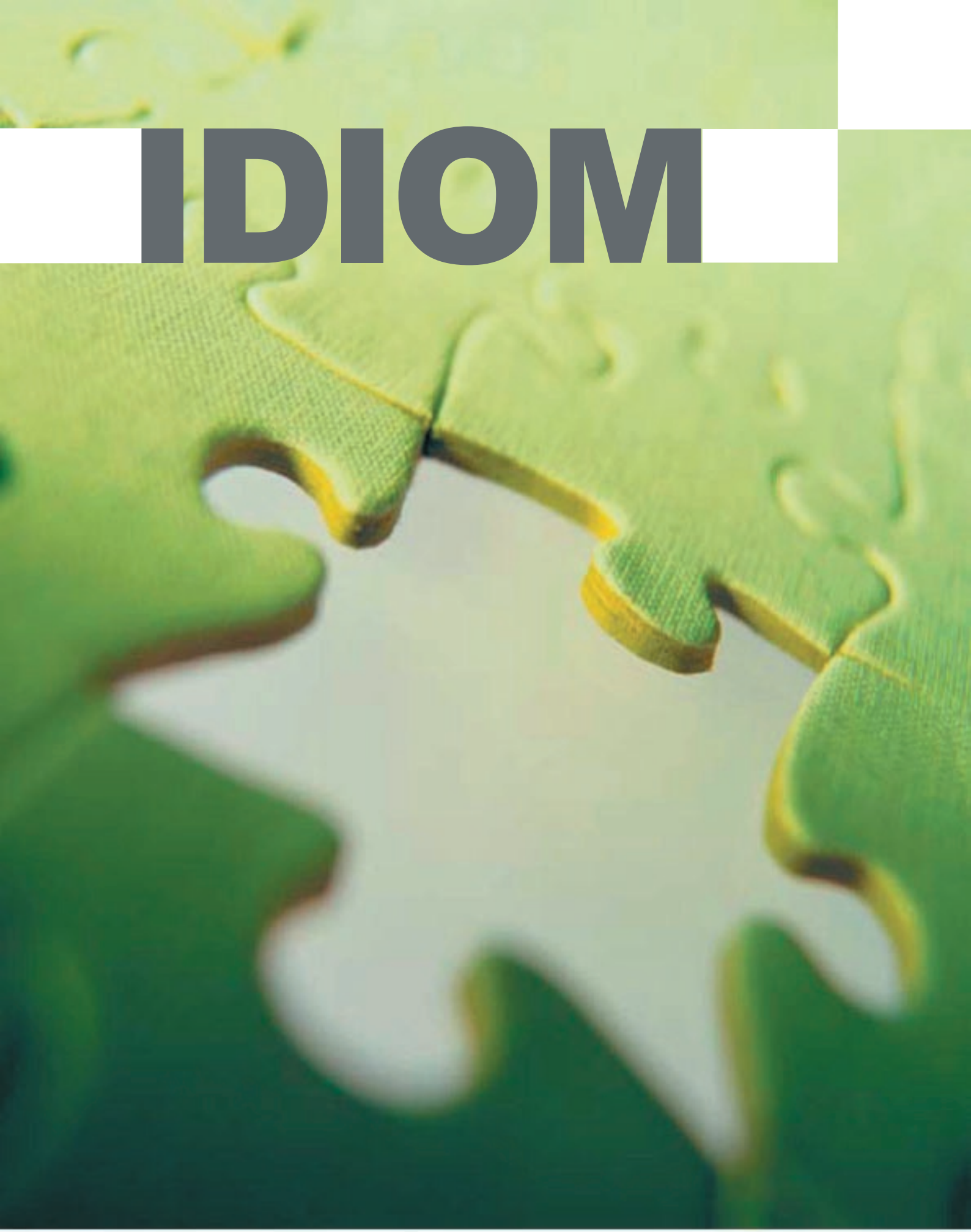
IDIOM



IDIOM + BUSINESS RULES BEANS

Overview

One of the key features of the IDIOM Decision Suite is the range of options available for deployment of the IDIOM decisions. IDIOM generates simple components that can be integrated into most common architectures. With the release of the recent version of Websphere Enterprise Edition, IBM has added a framework for Business Rule Beans. This white paper illustrates how IDIOM and the Business Rule Beans can be integrated within the IBM Websphere environment to offer a comprehensive rules management strategy.

Websphere Business Rule Bean Summary

The Business Rule Beans framework extends the scope of the Websphere Application Server Enterprise Edition to support business applications that externalise their business rules. Business Rule Beans are designed to remove the volatile components of a system (i.e. the business rules) from the application to allow changes to be made to the application without touching the applications system components (user interfaces, database interfaces, business object structure).

Business Rule Beans are implemented as standard Java components (Java Beans or Enterprise Java Beans) which are managed by the Websphere environment (much like an EJB component). To access the Business Rule Bean a program simply needs to implement a trigger point; this trigger point interfaces with the Business Rule Bean framework to execute the business rule that is encapsulated within a particular Business Rule Bean.

Why use IDIOM with Business Rule Beans?

While IDIOM and Business Rule Beans are trying to solve the same basic problem, the two technologies have approached this problem from very different perspectives. IDIOM has focused on the external management of business rules, and the design and development of these rules using an application that is targeted at the business user. Business Rule Beans, on the other hand, has addressed the infrastructure challenges of removing business rules from an application and accessing these rules via an execution framework.

The difference in approach makes the two approaches complementary. IDIOM was specifically designed to make no assumptions on the runtime environment (the system). The IDIOM decisions can be deployed within an Enterprise Java Bean, at the end of a messaging infrastructure, as a web service, or as a direct interface to a Java object. Websphere Business Rule Beans offer a deployment approach that will allow all business rules (whether managed by IDIOM or implemented directly within a Business Rule Bean) to be represented to an application through a common framework.

IDIOM was not designed to manage all business rules. The impetus behind IDIOM was to manage the business rules that a business user owned and understood. So, IDIOM is intended to manage rules such as the discount loading for a type of client, or the approval of an insurance contract, but not to provide a common validation lookup for an account number. As a consequence, an application will need to implement various rules that are not managed by IDIOM - the combination of IDIOM and Business Rule Beans ensures that all business rules will be executed via a common framework.

How to use IDIOM with Business Rule Beans

When an IDIOM repository is generated for release into a target environment, a package of objects is generated. These objects need to be deployed and integrated into an application. At execution time, instructions are given to the IDIOM DecisionServer object indicating which business rules need to be executed. This section will discuss an approach for the encapsulation of the IDIOM generated objects within a Business Rule Bean implementation, and outline the options available for configuring the Business Rule Bean to execute sets of rules within IDIOM.

Runtime interface to IDIOM

The runtime execution of rules within IDIOM is accessed via a single class called the DecisionServer. This class provides a public interface to the decisions and rules that have been defined within the IDIOM environment. The interface to this object is very simple, as illustrated in the following method definitions:

[Note: the code samples may be abbreviated for clarity]

```
public static void executeDecision(Date effectiveDate,
                    List documents,
                    String IdiomSystem,
                    String IdiomScope,
                    String slot,
                    String decisionName)

public static void executeScope(Date effectiveDate,
                    List documents,
                    String IdiomSystem,
                   String IdiomScope)

public static void executeSlot(Date effectiveDate,
                    List documents,
                    String IdiomSystem,
                    String IdiomScope,
                    String slot)
public static void executeSlots(Date effectiveDate,
                    List documents,
                    String IdiomSystem,
                    String IdiomScope,
                    String startSlot,
                    String endSlot)
```

This interface requires you to be able to specify values for the IDIOM environment, such as the system and the scope, in addition to the decision groups, or the individual decision, that you want executed.

Business Rule Bean Components

The Business Rule Bean framework requires that a rule component implement the interfaces defined within an IBM Rule Implementor interface (com.ibm.websphere.brb.RuleImplementor). This interface defines the contract that a business decision will have with the Websphere environment and the Business Rule Bean framework. This interface is illustrated below.

```
public Object fire (Trigger Point tp,
                    Object target,
                    IRuleCopy rule,
                    Object[] firingParms)

public String getDescription()

public void init(Object[] initParams,
                    String[] dependentRules,
                    String userDefinedData,
                    IRuleCopy rule)
```

Once implemented, a Business Rule Bean is deployed into the environment and the implementing class is associated with a name (JNDI name). The calling application will reference the Business Rule Bean by this name and interface with it via a trigger point (examples below).

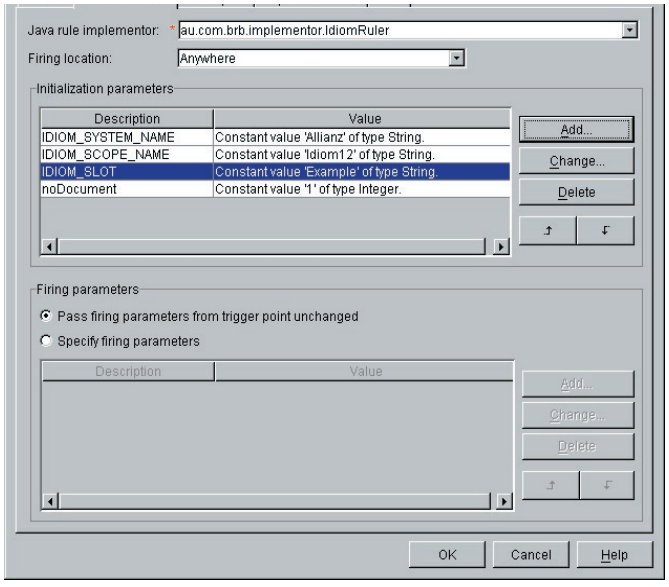
An IDIOM Business Rule Bean

The following code illustrates the integration of IDIOM within a Business Rule Bean. In this example the configuration of which rules (decisions in IDIOM terminology) will be executed is externalised from both the rule and the application calling the rule, and is managed by the deployment descriptor of the rule itself. The benefit of this approach is that a single IDIOM decision can be deployed and configured as a rule many times, each deployment having a specific binding to decisions or groups of decisions within IDIOM. This behaviour can be managed by the deployment configuration with no code changes required.

```
public class IdiomRuler implements RuleImplementor, VehicleConstants {

   //These variables are initialized in init()
   public String IDIOM_SYSTEM_NAME;
   public String IDIOM_SCOPE_NAME;
   public String IDIOM_SLOT;
   public int noDocuments;

   //set to true at the end of init
   boolean initialized = false;

   public IdiomRuler() {
      super();
   }

   /**
       Executes a call to the IDIOM DecsionManager to execute specific
Decisions/rules
    */
   public Object fire(TriggerPoint tp,
                Object target,
                IRuleCopy rule,
                Object[] parms) throws BusinessRuleBeansException {

      Boolean retV = new Boolean(true);

      List list = new ArrayList();
      if (initialized) {
         // One parameter is expected - the Idiom file
         ImplementorHelper.assertParamLength(parms,
                              NoDocuments,
                              "IdiomRuler.fire");

         //Populate a list of documents from the
         //rule parameters - the number of documents
         //is defined in the initialisation
         list.clear();
         for(int I = 0; I < noDocuments; I++){
            list.add((Document) parms[I]);
         }

         try {
            //execute the decisions on documents provided via
            //the parameters and the configuration as specified
            //in the initialisation
            DecisionServer.executeSlot(new Date(),
                              list,
                              IDIOM_SYSTEM_NAME,
```

```
                        IDIOM_SCOPE_NAME,
                        IDIOM_SLOT);
        }
        catch (Exception e) {
           retV = new Boolean(false);
        }
     }
     return retV;
  }

  /**
   This method is called by the framework and initialises the
   Rule based on the configuration defined during deployment
   */
  public void init(Object[] parms,
              String[] dependentRules,
              String userDefinedData,
              IRuleCopy rule) throws BusinessRuleBeansException {

     if (parms == null || parms.length == 0)
        return;

     // 4 parameters are expected
     ImplementorHelper.assertParamLength(parms, 4, "IdiomRuler.init");
     // Store the initialization parameters.
     IDIOM_SYSTEM_NAME = (String) parms[0];
     IDIOM_SCOPE_NAME = (String) parms[1];
     IDIOM_SLOT = (String) parms[2];
     noDocuments = Integer.parseInt((String) params[3]);

     Initialized = true;
  }
}
```

The Business Rule Bean configurator, as illustrated in the following image, manages the configuration of the rule and sets the values for the initialisation parameters. These parameters allow the Business Rule Bean to be configured to interface with IDIOM as illustrated in the example above.



The calling program

After the Business Rule Bean has been implemented and deployed into the Websphere application server, the calling application needs to implement a trigger point and fire a rule (using rules JNDI name).

```
……

// call the IDIOM Business Rule Bean to execute decisions
TriggerPoint tp = new TriggerPoint();
//populate the rule parameter with the appropriate document(s)
//for IDIOM
Object[] parms = {vehicleFormBean.getDocument()};

tp.disableCaching();
tp.setCombiningStrategy(CombiningStrategy.RETURN_FIRST,
            TriggerPoint.ALL_RULES);

Object returnObject = tp.trigger(null,
                  Parms,
                  "Idiom/Allianz/vehicle1");
……

//continue with application logic
//display pages, inspect document etc
```

If the IDIOM Business Rule Bean had been configured to interface to different groups of decisions, or even to execute a specific decision within IDIOM, it would have been deployed with a different JNDI name. This name is used in the calling program to indicate which rule will be 'fired' by the Business Rule Bean framework.

Conclusion

The combination of IDIOM and the IBM Business Rule Beans offers a common framework for the management of a wide range of business rules. Both the Business Rule Bean framework and IDIOM are designed to assist in externalising business rules and decisions from application code. IDIOM in particular is designed to offer business owners an environment in which they can define and manage their core business rules, while the Business Rule Bean framework provides an environment in which these rules are isolated from core system code.

The deployment of IDIOM within the Business Rule Bean framework allows the IDIOM components to be encapsulated within a common rules management framework. This framework is capable of delegating to IDIOM and is also capable of managing those rules that are not implemented in IDIOM. The packaging of the IDIOM runtime components within a common framework ensures that applications can be developed against

INVESTORS GUARANTY GLOBAL ALLIANCE

a standard architecture and framework, while the rules can be managed externally to the application.

The Business Rule Beans framework offers many alternatives for encapsulating the IDIOM runtime components. A Business Rule Bean could encapsulate a single decision, a group of decisions, or all decisions for a business domain (defined in IDIOM by a Scope). This flexibility allows for application developers to be further removed from the rule components during application development and supports the deployment configuration of business rules.