IDIOM

Web Services and IDIOM

Overview and Background

One of the key features of IDIOM is the flexibility enabled in the runtime deployment of the IDIOM decisions and formulas. IDIOM generates simple components that can be integrated into most common architectures. This whitepaper seeks to explore options for the deployment of the IDIOM decisions and formulas as web services.

Terminology

References are made in this document to the following IDIOM specific terms:

Scope, Decision and Decision Group. These concepts are clearly described in the White Paper `Introduction to IDIOM Global'. However, a brief outline is also provided below:

A SCOPE is the specification of the set of documents required to execute a set of Decisions. The scope therefore implicitly identifies a set of decisions for a business domain, and is the only required specification parameter when invoking IDIOM.

DECISIONS are specific business defined outcomes that are captured within the documents that are identified by the scope. For instance, a decision may determine a status of "Approved" for an insurance policy; or, its price.

A DECISION GROUP is a set of decisions that collectively address a business problem. For example, validation may involve many decisions, in which case all of the 'validation decisions' can be addressed as a group.

Why a web services deployment?

Web services offers another deployment option for IDIOM in addition to direct invocation and messaging. An implementation of a web service is provided with IDIOM (described in later sections) that packages the standard IDIOM execution interface as a web service. The remainder of this document will discuss this and other approaches to web service integration with IDIOM.

Before dealing with the technical approaches to web service integration with IDIOM, it is worthwhile looking at factors that are likely to influence the decisions to package IDIOM as a web service.

Internal Integration - typically an organisation finds that business rules are distributed across many systems and are implemented in many languages. IDIOM offers organisations the ability to manage repositories of decisions and to remove the management and execution of decisions from a wide variety of systems. IDIOM supports a runtime implementation for both Java and .NET, however it is understood that there are many other languages supporting a wide variety of business decisions. We see the encapsulation of IDIOM in a web service as an option to manage the execution of decisions in systems in which IDIOM does not have a supported runtime. This decoupling of decisions and the application via a web service allows IDIOM to be used in a much more diverse environment.

Externalisation of decisions and rules - the management and control of business rules independently to systems is not purely an internal problem. Many organisations package and resell their products/services through intermediaries; these intermediaries often implement their own systems to support the distribution of these services (e.g. banks may sell insurance products on behalf of the insurer and implement systems that manage the sales and distribution of these products). Once the core business decisions are implemented in IDIOM, the web services environment offers organisations the ability to offer a programmatic (non-GUI) interface (i.e. a web services interface) to partner organisations. This allows the business decisions to remain within an organisation but to be exposed to partners, facilitating existing relationships.

How to deploy IDIOM within a web service

IDIOM Standard Web Service Interface (wsdl)
The standard web service interface provided with IDIOM offers an alternative to the direct invocation of the decision server components (either via .NET or JAVA). The interface described below provides a low-level interface to IDIOM and is appropriate for an internal deployment of IDIOM as a web service. This web service provides a fine-grained RPC-style interface rather than a business-oriented interface and is intended to replace the direct call to IDIOM.

This web service interface could be used if a requesting component was developed in a technology that cannot directly integrate with either of the two supported deployment platforms for IDIOM (for example a 4GL or COBOL). Alternatively,

a message-oriented (e.g. MQ Series or JMS) or CORBA interface could be used. The decision on which integration method to use will depend on the available technology and on the desired quality attributes of the solution.

The web service interface to the IDIOM Decision Engine could also be used when multiple applications are using a single repository of IDIOM decisions and there is a desire to manage the runtime execution of these rules in a single component (hence, a web service).

The web service packaged with IDIOM accepts the configuration details for a given execution as regular message parameters. However, it makes no reference to the XML documents (business objects) to which the decisions will be applied. The XML documents should be added to the message as an attachment using the techniques described in the SOAP with attachments specification. In the current version of WSDL (version 1.2), there is no capacity to describe document attachments - there is therefore an expectation that the business objects will be passed as XML documents to the web service (this cannot be communicated formally with the current WSDL specification).

```
...
<types>
    <complexType name="decision"/>
        <sequence>
        <element name="first" type="xsd:string"/>
        <element name="last" type="xsd:string"/>
        </sequence>
    </complexType>
</types>

<message name="executionConfiguration">
    <part name="repositoryName" type="xsd:string"/>
    <part name="scopeName" type="xsd:string"/>
    <part name="decisions" type="decisions"/>
    <part name="effectiveDate" type="xsd:string"/>
</message>

<operation name="executeDecisions">
  <input message="executionConfiguration"/>
</operation>
...
```

Encapsulated IDIOM web service interface (WSDL)
Another approach to interfacing with the IDIOM Decision Engine via a web service is to encapsulate the IDIOM Decision Engine within a customised web services facade. Several of the reasons for doing this are summarised in the list below.

❑ The internal structure of an IDIOM Decision Engine does not need to be exposed.
❑ The internal and external representations of the business objects need to be isolated from each other

❑ A business interface needs to be presented to the requestor of the web service.
❑ Authentication or authorisation needs to be undertaken prior to executing the IDIOM rules.
❑ The IDIOM Decision Engine manages a large number of decisions, and these decisions need to be presented via a more coarse-grained interface.

Example Scenario

An insurance company is using IDIOM to manage the rules and decisions related to one of its products. It has defined a common set of business objects (expressed as XML schemas) and has defined rules relating to the product within IDIOM. The rules/decisions that the insurance company has chosen to manage within IDIOM are categorised as follows

❑ Underwriting decisions (approval for new business)
❑ Rating decisions (determine basic pricing)
❑ Discount/Commission decisions (determines discounts etc. based on the intermediary processing the business)
❑ Claim assessment (approve/decline a claim)

The insurance company has decided to provide two mechanisms to access these rules - a web application for customers to apply for insurance online, and a web service for insurance intermediaries (for instance, a broker within the sales channel) to implement system-system interfaces to access the decisions and rules. The following section outlines two approaches to implementing the web service interface to allow the insurance intermediaries access to the rules managed by IDIOM.

Business Object Attachments

The example below illustrates a web service implementation for the aforementioned scenario. In this approach, the business objects are passed to the web service as attachments to the SOAP message (as with the standard web service provided with IDIOM). The details of the repository configuration have been hidden from the web service user and have been repackaged as business operations. It is then the responsibility of the web service itself to manage the mapping between the operation and the repository configuration (i.e. the mapping between operations and scope, decision group, decisions etc). This approach follows the discussion on document centric computing for web services on the W3C website :
http://www.w3.org/TR/ws-desc-usecases/#document-centric-computing
.

```
.....
<!-- execution date is provided to allow the historical assessment of insurance
claims etc - >
<message name="executionConfiguration">
  <part name="executionDate" type="xsd:string"/>
  <part name="brokerID" type="xsd:string"/>
  <part name="brokerAuthenticationToken" type="xsd:string"/>
  <message>

<operation name="underwriteProduct">
  <input message="executionConfiguration"/>
</operation>
<operation name="rateAndPrice">
  <input message=executionConfiguration"/>
</operation>
<operation name="calculateCommissionAndDiscount">
  <input message="executionConfiguration"/>
</operation>
<operation name="assessClaim">
  <input message="executionConfiguration"/>
</operation>
.....
```

Type safe messages

In the following example, the business objects themselves are passed through the web services interface within the messages (as opposed to the document approach above). The business objects themselves have been defined within the DomainDef.wsdl file and will represent the same structure as the business objects required by IDIOM. This coupling is achieved through having a single definition of the domain (an XML schema) that is used by both IDIOM and the web service interfaces.

One of the principles of IDIOM is that the business objects may have their state altered as a result of the execution of decisions; to reflect this, the input and output messages contain the business objects.

This approach, while being type safe and ensuring that the format of the message is exactly as IDIOM expects (through the use of a shared definition of the business objects), does require more work on behalf of the caller. The nature of the business object definitions is that they are quite often large. This means that the web service interface (in particular the message) is likely to be complex and will require the caller to populate a SOAP message with a complex XML object.

```
......
<definitions targetNamespace="http://www.example.com/insurance"
        xmlns:idm="http://www.example.com/services/idiom"/>

<import namespace="http://www.example.com/services/idiom"
     location="http://www.example.com/services/domain/DomainDef.wdsl/>
```

```
<message name="insuranceProductInputMessage">
  <part name="executionDate" type="xsd:string"/>
  <part name="broker" type="idm:Broker"/>
  <part name="customer" type="idm:Customer"/>
  <part name="risk" type="idm:Risk"/>
  <part name="premium" type="idm:Premium/>
<message>

<message name="insuranceProductOutputMessage">
  <part name="broker" type="idm:Broker"/>
  <part name="customer" type="idm:Customer"/>
  <part name="risk" type="idm:Risk"/>
  <part name="premium" type="idm:Premium"/>
  <part name="error" type="idm:ExecutionErrorContainer"/>
<message>

<operation name="underwriteProduct">
  <input message=" insuranceProductInputMessage "/>
  <output message=" insuranceProductOutputMessage "/>
</operation>
<operation name="rateAndPrice">
  <input message=" insuranceProductInputMessage "/>
  <output message=" insuranceProductOutputMessage "/>
</operation>
```

Web Services Architecture

No matter which style of message is used when deploying IDIOM in a web service environment, it is recommended that the service be designed using a web services facade architecture. A block diagram providing a summary of this architecture is shown in Figure 1. The key components of the web services facade architecture are:

❑ SOAP interface – it provides a SOAP compliant interface to the underlying facade

❑ Web Services facade – this provides the coarse-grained, business-oriented interface that client applications interact with

❑ Object Assembler – responsible for transforming data from an external representation to the internal representation used by the Decision Suite.

❑ Session Manager – manages any inter-invocation state that may be required

❑ Orchestration Manager – responsible for managing the workflow and routing within the facade. Can use IDIOM decisions to manage the workflow

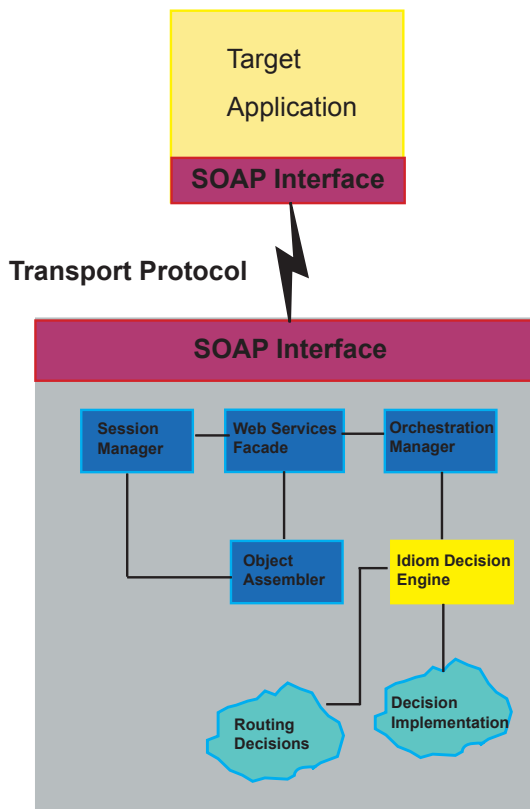❑ IDIOM – the IDIOM Decision Suite runtime component and generated decision code.

Figure 1 IDIOM in Web Services Facade architecture

Conclusion

We have seen how IDIOM can be deployed in a web services environment in the following ways

- ❑ IDIOM standard web service
- ❑ Document-centric web service
- ❑ Type safe message based web service

The decision on which approach is best depends on the individual requirements of a particular environment. Each approach has advantages and disadvantages. Web service encapsulation of IDIOM is very easy; IDIOM was designed to operate in an XML centric environment and it is therefore well suited to the web services environment. Whether the business objects are passed as documents or in messages does not affect IDIOM itself as these business objects will be XML and will conform to the definition of the business object that IDIOM expects.