Legal Notices

Third Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

.

Introduction

This architecture guide is intended to give an overview of the OIPA application and present the key benefits of using the application.

Customer Support

If you have any questions about the installation or use of our products, please visit the My Oracle Support website: https://support.oracle.com, or call (800) 223-1711.

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Product Overview

Oracle Insurance Policy Administration (OIPA) is an adaptive, rules-based policy administration system that provides full record keeping and support for all policy lifecycle transactions (such as policy issue, billing, collections, policy processing and claims). With Oracle's policy administration system, insurers can rapidly adapt to changing business needs and regulatory requirements while supporting straight-through processing throughout the policy lifecycle.

OIPA is used by leading insurers globally to accelerate product development and speed up time to market for differentiated life insurance, unit-linked and annuity products. The system enables insurers to provide real-time policy servicing of customers and sales channels throughout the policy lifecycle for increased retention and loyalty. It also helps insurers reduce risk and support compliance, while better managing the business to optimize performance through use of a single system.

Accelerates Speed to Market for New and Adapted Products

Insurers require the ability to rapidly bring to market innovative products that stand out from the competition, capture more market share, and ultimately maximize profitability. They can no longer rely on inflexible, aging legacy systems that require heavy IT intervention and hamper their ability to quickly adapt to evolving market dynamics and regulatory conditions.

OIPA enables insurers to accelerate product development and time to market for differentiated life insurance, unit-linked and annuity products globally. The system's rules-driven configuration capabilities are unmatched in the industry. Almost all changes to the system—including products, fields, screens, languages, and currencies—can be made without ever touching the core code or recompiling the data base structure. The system does the heavy-lifting through a user-friendly Rules Palette visual configuration tool, pre-configured product examples, the ability to reuse rules and clone products, and release management.
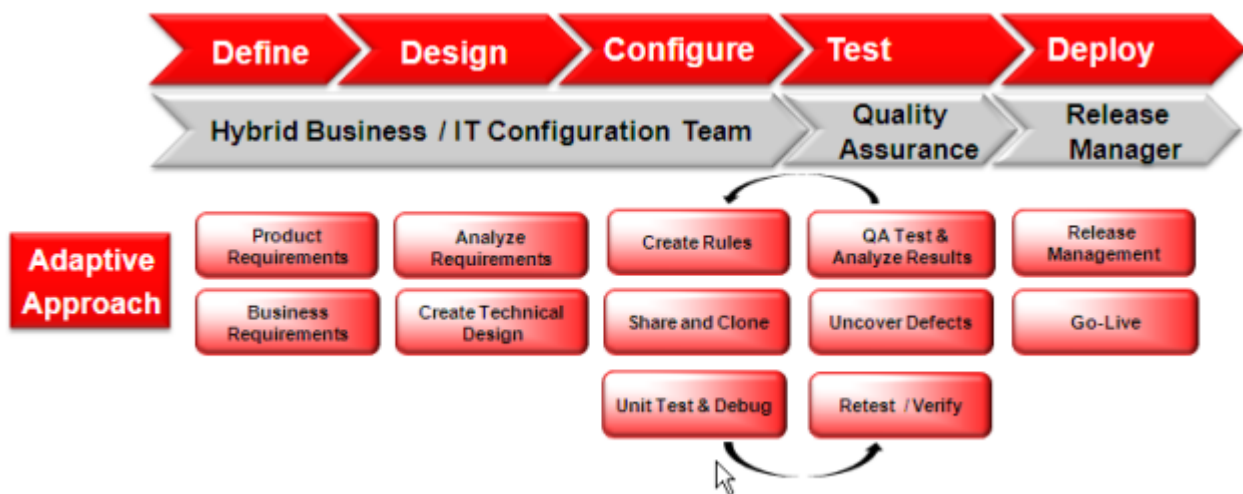
Improves Customers and Channel Loyalty through Better Servicing

Delivering better service to customers and sales channels throughout the policy lifecycle is critical to promote loyalty and retention. OIPA can help insurers improve servicing by enabling them to provide real-time access to policy information and the ability to process transactions or events through Web Services integration to self-service portals and other systems. It also provides a single view of the customer through full record keeping and support of all policy lifecycle transactions. Additionally, it can help insurers reduce manual processing by customer service representatives (CSRs) and drive consistency by automating support of business processes and validation of transactional information for improved customer servicing.

Reduces Risk While Better Managing the Business for Growth

It is not uncommon for insurers to have multiple, disparate policy administration systems supporting multiple lines of business, including closed blocks. These systems are often inflexible, hard coded, and expensive to maintain.

With Oracle Insurance Policy Administration for Life and Annuity, insurers can consolidate from multiple systems to a single platform to support life, health, unit-linked and annuity products, significantly reducing maintenance costs. Its highly extensible and open architecture also allows for easy integration with front and back-office systems so insurers can progressively renovate their application portfolio while avoiding "big-bang" system replacement.

Insurers can further mitigate their risk during conversions by leveraging proven best practices and a broad industry ecosystem including Oracle Insurance and technical resources and industry partners.

Eases Integration with Other Systems

The open, standards-based architecture of OIPA allows for integration with other third-party systems such as illustrations, new business and underwriting, claims, billing, enterprise document automation / customer communication management, rating, and more. In addition, OIPA is compatible with Oracle Application Integration Architecture (AIA), which enables rapid, low-risk integration with other back-office systems and existing legacy systems.

OIPA also integrates with Oracle's service automation, enterprise document automation and content management solutions. This gives customer service representatives a 360-degree view of the customer, including associated documents, correspondence, confirmations, statements, and policy data—increasing productivity and customer satisfaction.

Key Features and Benefits of OIPA solutions

Line-of-business and Product Agnostic

OIPA is highly configurable through XML business rules and designed to be both line of business and product agnostic. It enables insurers to create truly innovative products and riders, from simple to complex, across life, health, unit-linked and annuity products.

Web-based, Modern, Highly Extensible

OIPA is a modern system that is built on open, J2EE-based architecture and can be deployed across numerous technology stacks. The system is Service-Oriented Architecture (SOA)-enabled allowing integration through Web Services with other insurance systems and Web portals to support straight-through processing. The system's browser-based User Interface further promotes ease-of-use by both business and IT users.

Unparalleled, Flexible Rules Configuration

The OIPA streamlined architecture separates the rules, which support business and product logic, from the base code, minimizing the need for heavy IT intervention during the configuration process. Its highly flexibly rules-con-figuration capability empowers business and technical users to collaboratively configure changes using business rules without the need to customize the system's core code or database structure. This helps shorten product development time cycles, while reducing the cost of configuring products and installing upgrades.

Another key benefit is the ability to reuse rules. Users configure rules once and reuse again for other products reducing development, testing, and maintenance.
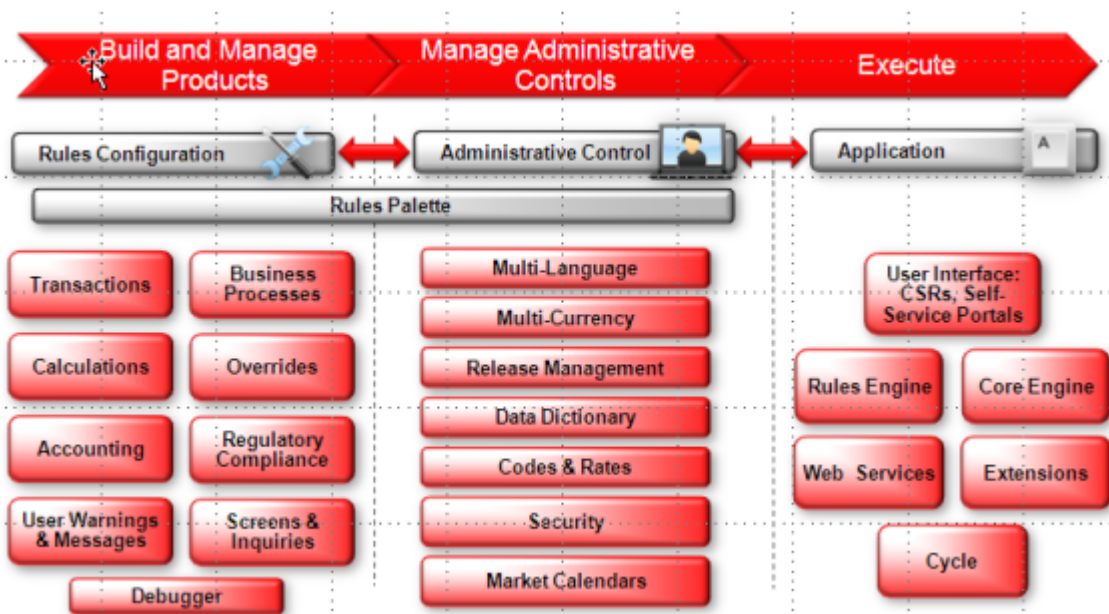
Oracle Insurance Policy Administration for Life, Unit Linking and Annuity is structured to provide competitive advantage through rules-based configuration separated from the base code and data structure.

Powerful Rules Engine

OIPA's rules engine allows business and technical analysts to configure transactions using business rules to support unique and creative product features. For example, an insurer may want to execute a distinct transaction on a policy, such as paying an agent a bonus on every sale. This innovation makes the insurer more attractive to agents and producers. It also promotes consistency and accuracy, reducing the time required for development, testing, and maintenance. Through rules-based configuration, business and technical analysts can create compliant calculations and reusable functions, such as tax withholding, to support state-filed products across life, health and annuities.

Rules Palette

The Rules Palette, a visual configuration tool with drag-and-drop functionality, simplifies rule creation, modification, and debugging. It offers unparalleled flexibility by enabling business and technical analysts involved in configuration to make changes by products, fields, screens, languages, currencies, and more through business rules—while facilitating improved collaboration with business and IT. With it, users can control security rights, rates, funds, and other items through administration tables for a centralized view of fixed and variable data for enhanced flexibility. The integrated Data Dictionary drives a consistent set of field labels and math variables.

Integrated Debugger

OIPA includes an integrated debugger tool within the Rules Palette. The debugger provides full exposure and step-by-step execution of formulas and complex calculations within policy examples (for example, a partial surrender charge or taxable gain). The ability to validate calculations and formulas contributes to reusability and reduced development cycles, testing, and maintenance.

Pre-configured Product Examples

OIPA includes pre-configured product configuration examples for guaranteed level premium term life, variable deferred annuity (without annuitization payout), and unit-linked fund processing. The pre-configured examples help provide a jump start to insurers during the configuration process and may be adapted by an insurer based upon the specific product and /or business requirements.

Product Cloning

OIPA provides the ability for users to quickly "clone" a product and reconfigure it to create a new one, resulting in accelerated speed to market. Users may clone rules from an existing product and reconfigure based on new requirement (for example, add a new secondary guarantee to a life or annuity product).

Corrective Processing

Corrective processing functionality is inherent within the system and is triggered by policy transaction reversals or compliant back-dated transactions. This eliminates the need for manual processing by customer service representatives (CSRs). In contrast, legacy systems are often high-touch and time consuming for CSRs to complete corrective undo and redo processing of policy transactions reversals.

Complete Traceability of Data

OIPA enables insurers to enforce compliance and best practices, reduce manual processing, and provide full visibility into transaction history throughout the policy lifecycle. It provides the ability for users to view the history at transaction and screen detail level to support compliance requirements and market conduct audits. This is available to CSRs or other employees as soon as the transaction has been processed.

Proven Performance – Tested Scalability

The system also is highly scalable to support the evolving business needs of the largest Tier One global insurers. Oracle recognizes that each implementation project is unique and specialized to each customer's requirements with system performance dependent upon the rules configured to support their products and lines of business. Performance tuning is an important phase of each project to meet client expectations.

Release Management

Release management capability with OIPA provides governance of rules migration from development, to testing, to the production environment throughout the product development lifecycle. This promotes consistency and further accelerates time to market for new and adapted products.

Globalization / Localization Support

OIPA includes several features to support internationalization and localization requirements of global and regional insurers, including support for multiple languages, locales and currencies in a single instance of the system.

OIPA Architecture Design Principles

The system architecture has been based on the following design principles that establish a consistent set of rules and guidelines for the design and development of the system:

1. The system architecture should have a multi-tier design with well-defined service layers to ensure flexibility and continued enhancement.
2. The application should be multi-platform, portable, and scalable.
3. The database access should be implemented in a consistent, database-independent manner.
4. The database traffic should be optimized to the utmost degree to increase scalability and availability of the system.
5. The application should support multiple locales, languages, and currencies in a single deployed instance.
6. The presentation layer should support, but not be limited to, a browser-based user interface.
7. The Shared Rules Engine should be implemented as a standalone component that is not dependent on any particular application.
8. The application's data model should be extensible by configuration to satisfy client-specific requirements.
9. The system should only implement generic business requirements and leave the client-specific requirement to the configuration.
10. The system should be extensible via extension points configured and implemented for client implementations.
11. The system should easily integrate with other technology components.
12. The system should leverage open standards wherever possible.

OIPA Key Concepts

Activity Processing

Activity processing is a fundamental part of the Oracle Insurance Policy Administration system. Almost every event that occurs in the insurance domain can be modeled as an activity in the system. An activity records all the changes it makes and provides the ability to undo any such changes. Activities are therefore fundamental to corrective processing in OIPA. Activities are transactional units of work, so they never leave a business entity in an invalid state.

The behavior of activities from capturing the input data to the resulting changes is configured in XML using the Rules Palette. The configured XML is called a transaction. An activity is an instance of a transaction.

Some typical OIPA transactions at the policy level are premium, billing and anniversary processing. Quite often as in policy administration, one event on a policy triggers another, such as a notification letter or recalculation. OIPA supports this by providing the ability to one or more activities as a result of processing an activity.
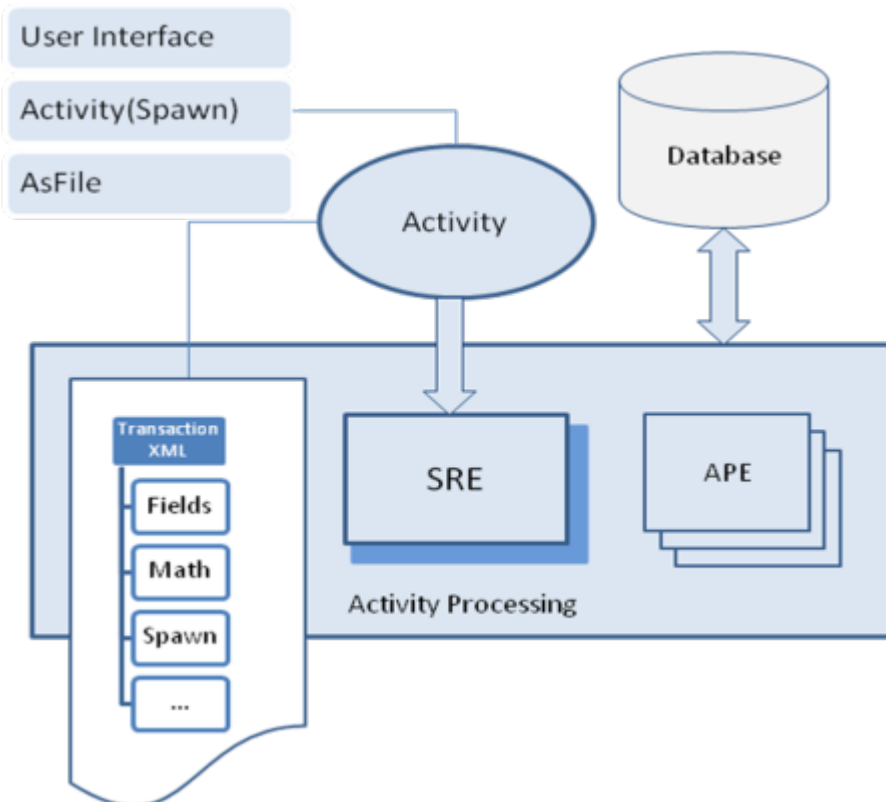
The above diagram illustrates the activity processing at a high level. The transaction XML configures input fields for the activities, the math to transform data, a set of rules to persist the changes, new activities that could be spawned, and so on.

An activity may be created by a user from the user interface, by another activity or as a result of incoming data from a web service. An activity is processed by the Shared Rules Engine, which is a component responsible for executing OIPA transactions and business rules. The results of activity processing are then stored in the database.

Screen Configuration

Screen Business Rules

Out of the box, the pages of the OIPA web-based user interface only implement generic functionality that is deemed to be of value to a cross-section of insurers. The pages can be customized through business rules to meet specific insurance product needs. The business rules are created and maintained by the Oracle Insurance Rules Palette in XML format and considered a part of a custom implementation, along with transactions and other configuration.

Business rules that govern the content, look and behavior of pages are called screen business rules. The screen rules provide a wide variety of features that allow customizing pages to meet client-specific needs.

A comprehensive set of input components: text, date, currency, drop-down combo-box, radio button, and so on.

» The ability to specify locale-specific field labels.

» The ability to set default field values that depend on the existing data, transformed, if needed, by complex math calculations.

» The ability to specify events and actions in response to user input: changing field values, disabling and enabling fields, showing and hiding fields, displaying field and page messages.

» Support for multiple languages, currencies, date formats, and so on.

» The ability to specify masks for input text depending on the user's security privileges.

» Validation of screen data triggered after data changes or by a page submission; alerting user with field and page validation messages.

As with other business rules, screen rules can be set up so that pages look and behave differently for different products and jurisdictions.
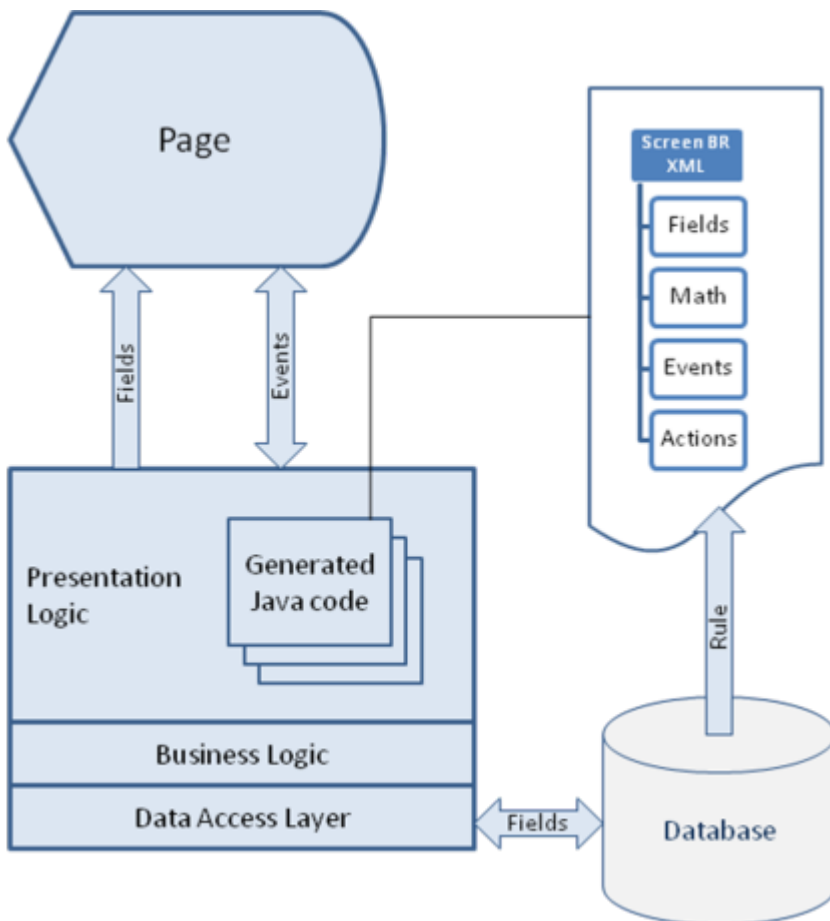
The above diagram illustrates the concept of the page configuration. A screen business rule is used to create a set of fields that are displayed on a page. A set of Java classes, generated at run-time and based on the rule's logic, is responsible for handling screen events – page load, page submit and field value changes.

Converting the rule logic from XML to Java classes does away with the inefficiencies of an interpreted language like XML and replaces it with the compiled efficiency of Java, improving system performance. Furthermore, when the system has to parse the same screen rule again, it recognizes the existence of a generated class and uses it, instead of regenerating the class.

When the user performs an action to save the information entered on the screen, the configured validation in the screen rule is invoked. Only after the screen rule configuration confirms the validity of the data does the system persist information to the database.

Configurable Dynamic Fields

Configuring the pages of the OIPA application with client-specific rules essentially customizes the application data model to satisfy customer's business requirements. The base application data model is extremely flexible and can be extended as required for a particular client implementation.

Only a few generic data fields, known as fixed fields and shared between all client-specific implementations, are stored by default in the application database. Fixed fields are represented as columns in the database tables. The dynamic fields that are configured through the screen business rules as described in the previous section are stored in the tables with names that have a suffix of Field (As<EntityName>Field). A value of a configured field of a business entity is stored in its own row in the corresponding Field table:

```
<EntityScreen>
  <Fields>
    <Field>
      <Name>TextField</Name>
      <DataType>Text</DataType>
    </Field>
    <Field>
      <Name>DateField</Name>
      <DataType>Date</DataType>
    </Field>
  </Fields>
</EntityScreen>
...
```

AsEntity (GUID,...)	
GUID1	Entity1
GUID2	Entity2
GUID3	Entity3

AsEntityField (GUID,Name,Type,Value,...)			
...			
GUID2	TextField	Text	...
GUID2	DateField	Date	...
...			
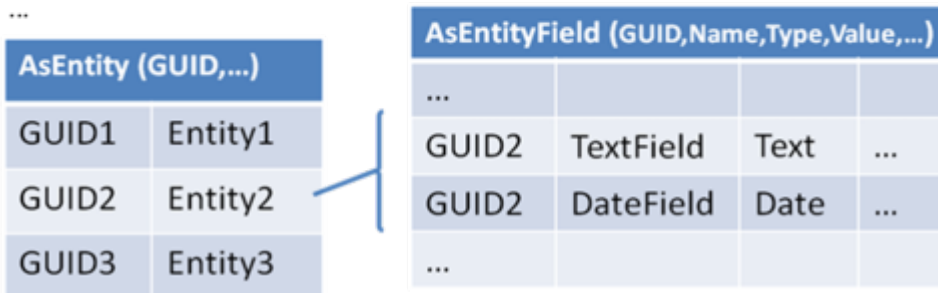
The above example shows a screen rule for a hypothetical business entity called Entity. Some business entity examples are Policy, Client, and Activity. The rule specifies that the Entity has two dynamic fields. The Entity's fixed attributes are stored in one row as column values in the AsEntity table. The AsEntityField table stores dynamic fields, one value per row, that match the configuration in the Entity screen business rule. Each row contains a GUID of a parent entity, field name, field data type, and value that is stored depending on field's data type.

This approach to data configuration and storage enables clients to extend the application data model without affecting the database schema. This not only makes database administration easier but also reduces the effort required to upgrade, because schema changes made in the core product with new releases will not conflict with changes made by clients.

Inbound Web Services

FileReceived

The OIPA application provides a web service called AsFile or FileReceived to electronically submit data to the system. The AsFile/FileFeceived web service allows an external system to send data in XML format to OIPA. Based on the client-specific configuration, the data can be transformed, validated, and inserted in the OIPA database followed by processing of related activities.
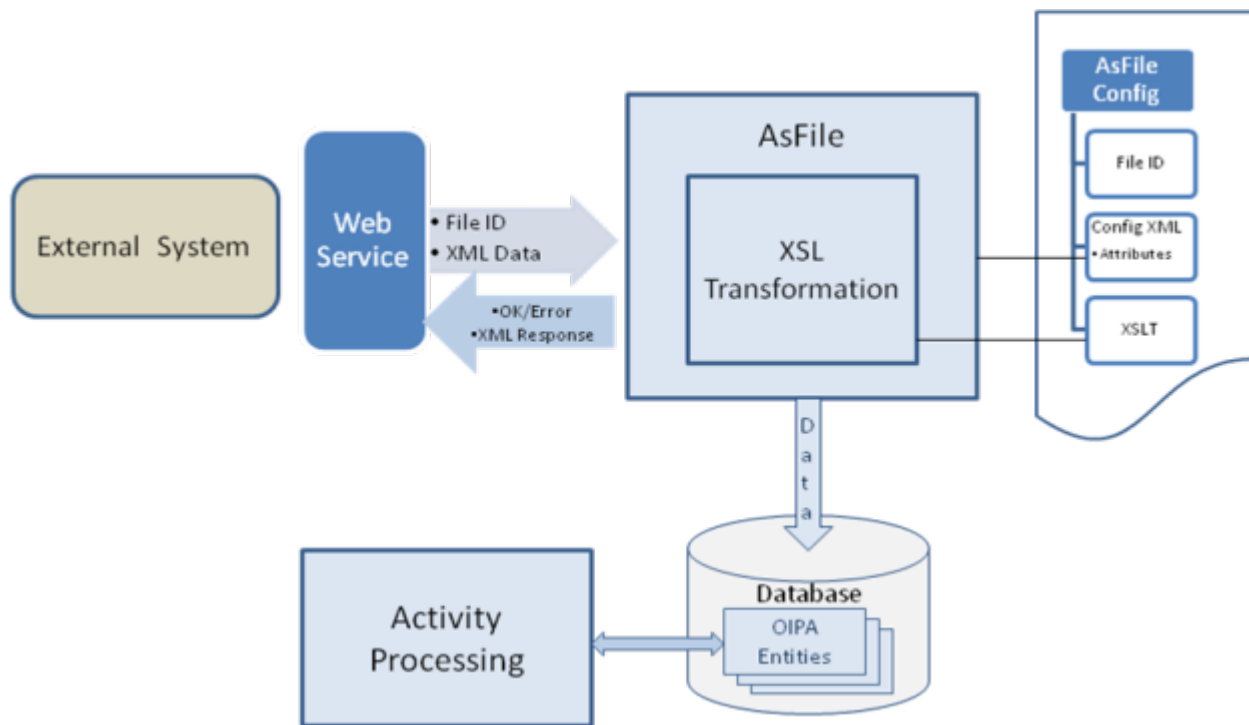
As the above diagram illustrates, a SOAP message is sent by an external system to the FileReceived web service and includes two parameters: FileID and XML data. FileID identifies the configuration from the AsFile table that will be used for processing and transforming inbound XML to OIPA business entities. Then, the created entities are persisted in the OIPA database. The AsFile configuration may also specify additional processing after the incoming date is stored in the database. Activities will be created and processed as dictated by business requirements for processing electronic submissions into the OIPA application.

At the end of the processing, a SOAP message is sent back to the external caller that includes the result of the processed request. If the processing has been successful, the outcome may also include an output XML constructed based on the provided configuration for the AsFile response.

Integration with external systems, including outgoing calls, is supported through extensions and is discussed later in this document.

DISQ

DISQ provides a simpler, more comprehensive interaction model than FileReceived. DISQ surfaces all of the data and associated operations in a consistent and flexible manner, facilitating any kind of service interaction.

While OIPA provides flexible services with AsFile and ExposedComputation, there is no formal XML schema that defines the input and output parameters for them. The payloads typically consist of a single string, which can be any kind of structured xml data. In contrast, DISQ provides a formal XML schema which describes the data and

operations that it exposes. DISQ requires no configuration in order for it to function, beyond normal configuration in the existing system.

A DISQ Request consists of the following parts:

» **Header** - provides a loose bucket of properties that can help influence the overall processing of the DISQ request.

» **Context** - identifies the resource that the DISQ request applies to. The Context section of the DISQ allows the caller to specify any combination of fields to uniquely identify the context for the Action.

» **Action** - identifies the operation to be performed within the context specified.

» **ResponseFormat** - identifies the information to be returned as a result of the action being performed.

```
<DisqRequest>
    <Header>
        <MessageId>671B973D-2AC7-4572-8E23-202FF4C5AB64</MessageId>       Header
        <Timestamp>2012-09-19T17:00:30.496-04:00</Timestamp>
    </Header>
    <Context>
        <Company>
            <CompanyName>Holding Company</CompanyName>                    Context = Holding Company
        </Company>
    </Context>
    <Get>
        <Client>
            <LastName>Hyatt</LastName>                                    Action = Get Client
        </Client>
    </Get>
    <ResponseFormat>
        <ResponseType>Default</ResponseType>                             Response = Only Fixed Fields
    </ResponseFormat>
</DisqRequest>
```

See the OIPA Web Services document for additional details.

Outbound Web Services

Outbound Services Connector (OSC)

The OSC leverages the extensibility framework to allow for outbound web service calls during math processing. The extension provides mechanisms to specify delivery targets and message templates. The latter allows users to map between the data at a given extension point and a downstream interface. Further, it avoids the need to develop individual extensions for each downstream interface, while maintaining application performance and throughput.

Cycle Processing

Cycle is a high-performance distributed subsystem of OIPA designed to process as many pending activities as possible in the shortest amount of time. Depending on the configuration of the pending activities, Cycle executes scheduled one-time and repeating insurance events. Cycle uses a distributed computing grid, concurrency techniques, multiple threads, automatic failover and scaling to deliver a robust batch processing solution.

In addition to processing pending activities, Cycle is also used to value insurance policies as frequently as specified by a customer. Typically, policy valuation is scheduled to be batch-processed by Cycle quarterly, semi-annually or annually.

In addition, Cycle is used to advance the current business date used in the processing of activities. The current business date is stored in the application database and is advanced to the next available business date.

Cycle Processing

Data Intake Processing

Data Intake is the process of receiving files from Group Customers for the purpose of importing data into the Oracle Insurance Policy Administration system (OIPA). The data in the files may result in many changes, including but not limited to the following:

» Adding a new member to the system
» Changing an Employee's elected coverage
» Adding a dependent to a coverage
» Enrolling a member and dependents
» Auto-cancelling coverage for a member
» Updating member information in the system

The insurance carrier's file receiving system will be responsible for handling incoming files that are submitted by Group Customers and prepare them for consumption by OIPA. Once the file is ready, the system will submit a request to OIPA to process a file, providing the unique file identifier.

Data Intake capability is built into OIPA Cycle. Each node that is receiving requests and processing files will be an instance of a cycle agent with Data Intake enabled. It uses the same proven pattern of Resumable and Runnable Tasks already implemented by Cycle and Scheduled Valuation.

System Architecture Overview

The OIPA system is implemented as a multi-tier J2EE-based server-side application. The presentation, business logic and data access have been developed and maintained as independent layers that run inside a single JVM. The Spring framework is used as a component container to assemble the components together and to access available services across the layers.

This is an architectural diagram of the OIPA application:
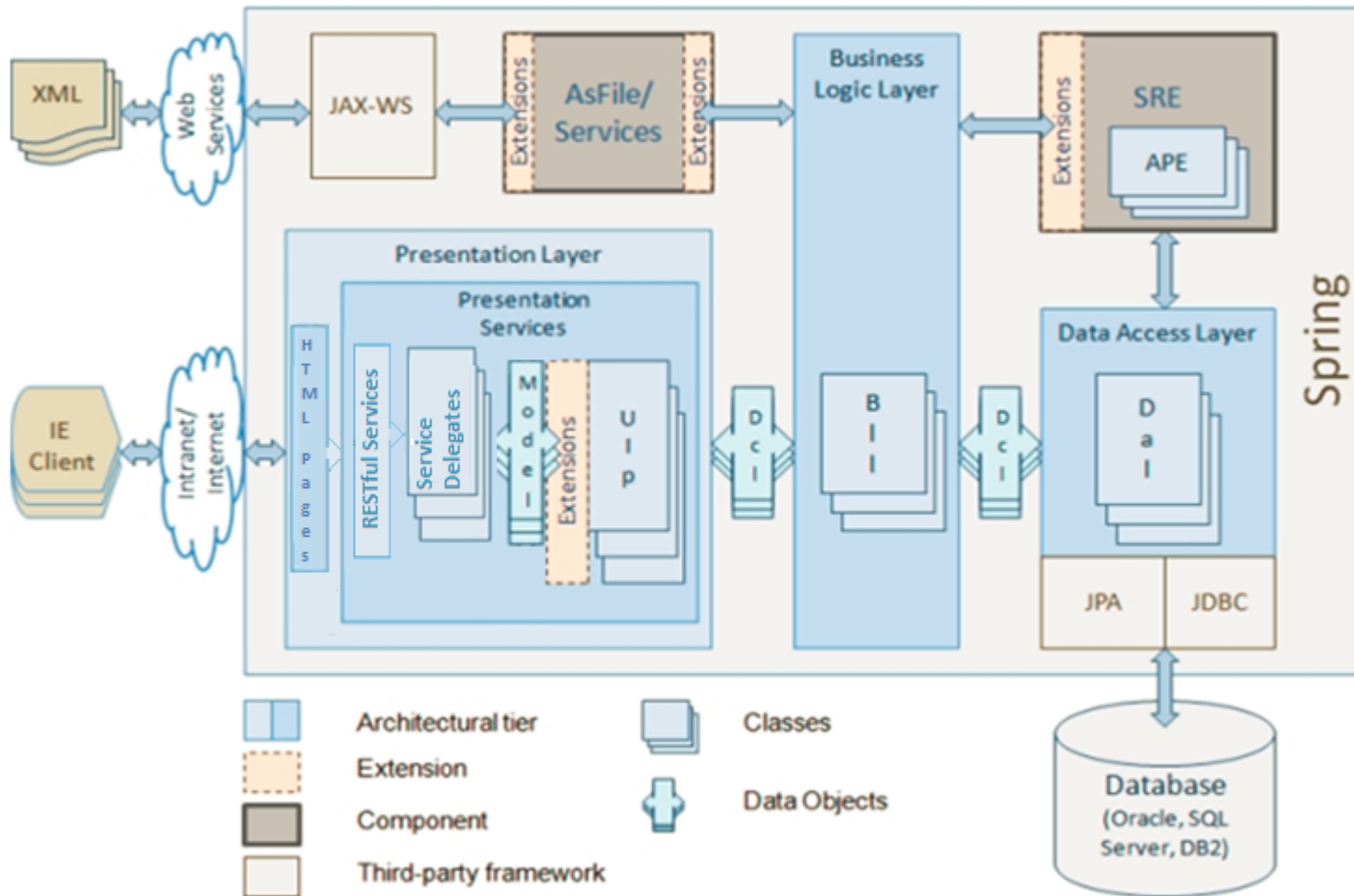
The following is the glossary of terms used in the diagram:

Data Access Layer (Dal) – an architectural layer that consists of Dal objects and which implements the data access, hiding the details of implementation within the tier.

Business Logic Layer (Bll) – an architectural layer composed of Bll objects that implements business logic as services available to the presentation layer, and also deals with some infrastructure aspects such as caching and transactional processing.

Data Carrier Layer (Dcl) – data carrier objects that carry data throughout the system.

Presentation Layer – an architectural layer responsible for implementation of the presentation logic; consists of the two tiers, HTML Pages and Presentation Services.

HTML Pages – Front pages using Oracle JET Framework. Oracle JET supports the Model-View-ViewModel (MVVM) architectural design pattern. In MVVM, the Model represents the application data, the View is the presentation of the data and the ViewModel exposes data from the Model to the view and maintains the application's state.

Presentation Services (UIP)– a technology-independent presentation tier that provides presentation services to an outer presentation tier.

RESTful Services – Service layer built using RESTful frameworks.

Service Delegates - Delegate layer which redirects service requests to appropriate pages

UIP - A Facade between User interface layer and Business Layer

Model – an object implemented for a user interface page that contains data displayed on the page; a data carrier between the two presentation tiers.

SRE (Shared Rules Engine) – a standalone component responsible for executing OIPA transactions and business rules

APE (Application Process Executor) – a class that implements a business rule executed within a transaction.

Extensions – configurable call interceptors that allow modification of the out-of-box functionality for client-specific implementations.

AsFile/Services – a component that exposes the business layer services as web services.

An application's data access details are hidden from the rest of the application inside of the Data Access layer. Java Persistence API is used to implement the data persistence for most of the data access requirements. Data access from the Shared Rules Engine that sometimes requires a greater degree of control over the generated SQL traffic is implemented using JDBC. The services provided by the Data Access layer are available through the Spring container and exchange data using the Data Carrier (Dcl) objects.

The Business Logic layer contains generic business logic shared between all custom implementations of the system that is complemented by the configured business rules. The exposed services include activity processing, accounting, allocations, and so on. The layer also deals with such infrastructure aspects of the application as transactional and grid processing, and caching. The services provided by the Business Logic layer are consumed by the presentation layer with the data exchanged through the Dcl objects, by the cycle agent for the batch processing, and also exposed through the web services to be used by external systems.

The Application UI is built on HTML5 using Oracle JET Framework.

The Shared Rules Engine is a separate and independent component responsible for processing activities and math configured in the OIPA business rules. The web services component that uses the Apache Axis framework allows configurable electronic submissions into OIPA and, in general, exposes OIPA business services to external systems.

The following is a list of the most significant frameworks and technologies used by OIPA:

» Spring framework as an application component container

» JPA as a primary method to access persistent data

» JDBC for accessing persistent data from SRE to ensure maximum data access performance

» Oracle JET framework to implement browser-based user interface

» JAX-WS and JAX-RS to implement web services

» Coherence cache for caching rarely changed persistent data

» Coherence processing pattern as a grid computing framework

Shared Rules Engine

The Shared Rules Engine (SRE) component performs activity processing in the OIPA application. Activity processing manages insurance events. SRE loads a transaction and processes the data according to the business rules and math associated with the transaction The transaction, business rules and actual insurance data are retrieved from the database.
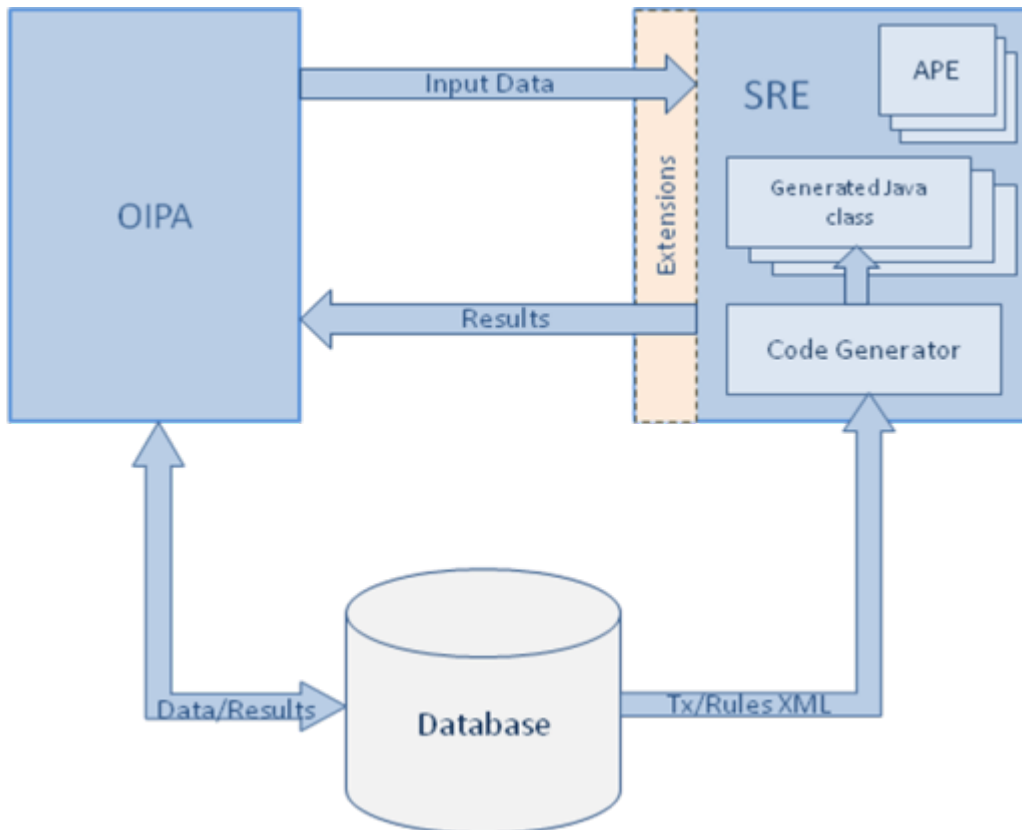
The above diagram shows a high level interaction between the calling application and SRE. OIPA calls SRE, provides input data, and implements interfaces to call back when additional date is needed by SRE. SRE does not directly make calls to the database, except for loading the transaction and rules associated with the current activity. When processing is complete, the results are packaged and returned to the calling application. Then, the results are committed to the database within a single database transaction.

It is important to note that the transaction XML is translated into a generated Java class that will be executed by SRE to ensure the best possible performance.

The following components of SRE come together in processing an activity:

» Processor
» Java code generator that includes a math translator
» Data access components implemented by OIPA to retrieve input data and persist results
» Application Process Executor (APE) business rules invoked by SRE
» SRE client-specific extensions

Configuration-based Code Generation

Most of the OIPA application business logic is configurable and contained in the business rules and transactions stored in the database in XML format. Some of these XML transactions and rules are executed during the batch processing, others from the browser-based user interface in real time. In either case, the performance is extremely important.

In order to provide the best possible performance, the XML rules and transactions are transformed by the Shared Rules Engine into generated Java classes that are then compiled and executed like any other Java classes in the application. There is no difference in the performance between executing business logic hard coded in the application by developers and when configured in XML by business analysts.

The process of code generation and compilation is expensive and is only done once, with the result being cached and used for all consecutive requests. The cache can distinguish between development and production application modes and detect when the business rule XML changes and requires re-generation of a corresponding Java class.

Security

Authentication

OIPA performs user authentication for both interactive users using Internet browser to access the system and web service calls. The users are prompted to provide a user name and password on the application's login page; these are then sent to the server. The web services are protected with the WS-Security that requires incoming web service calls to carry a security header with the user name and password. The password can be sent as a digest or as a text.

Both web service and user authentication is implemented through the same authentication service provided by the business logic tier of the OIPA application. The authentication service retrieves a matching user record from the OIPA database that contains basic user information and a secure digest of a password. The password digest is then compared to the digest of the incoming password and an authentication decision is made based on the result of the comparison. User records in the OIPA database are usually created by the Rules Palette.
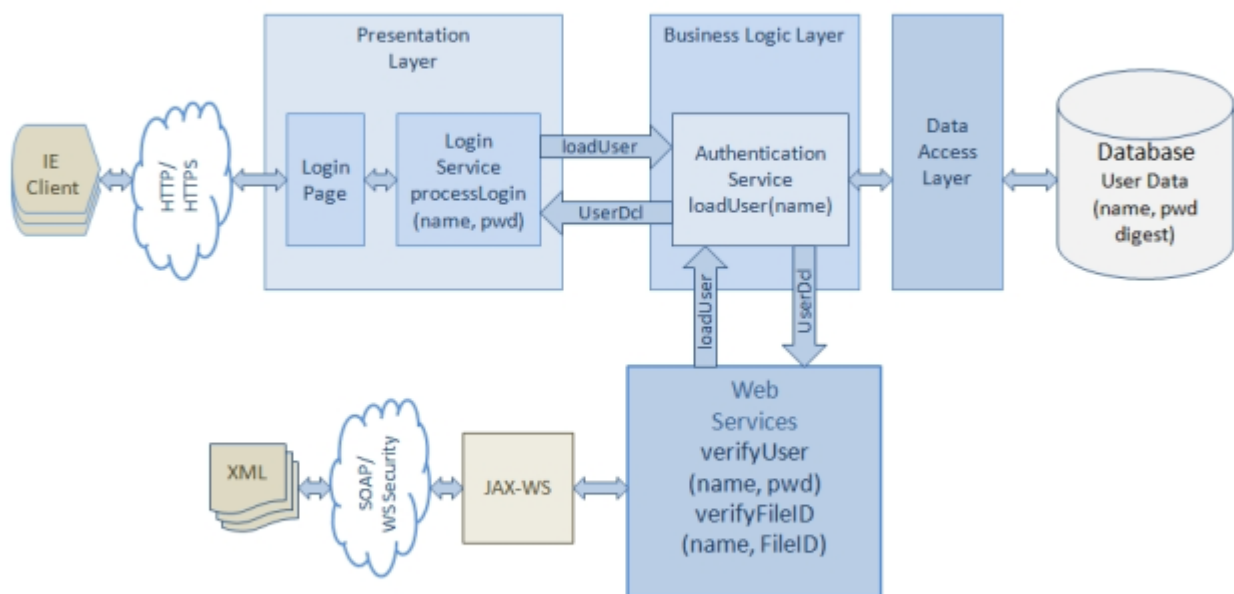
Using OIPA extensions, it is possible to implement alternative methods of user authentication to satisfy specific security requirements of a particular customer.

User Privileges and Role-Based Security

The OIPA user privileges and access restrictions implementation is based on the role-based access control (RBAC) model. According to this model, user permissions are assigned to specific roles or groups that are created for

various job functions. A user that is assigned particular roles, gains through those roles permissions to perform particular system functions. A user may belong to multiple groups that result in access granted to all resources authorized across the groups the user belongs to.

For example, users that are assigned to the CSR group (or have the CSR role) may not be able to execute such activities as issuing a policy or paying a death benefit. An Underwriter should be able to issue a policy. An administrator group is usually allowed access to all resources.

The following diagram shows what application resources are protected by the OIPA security:
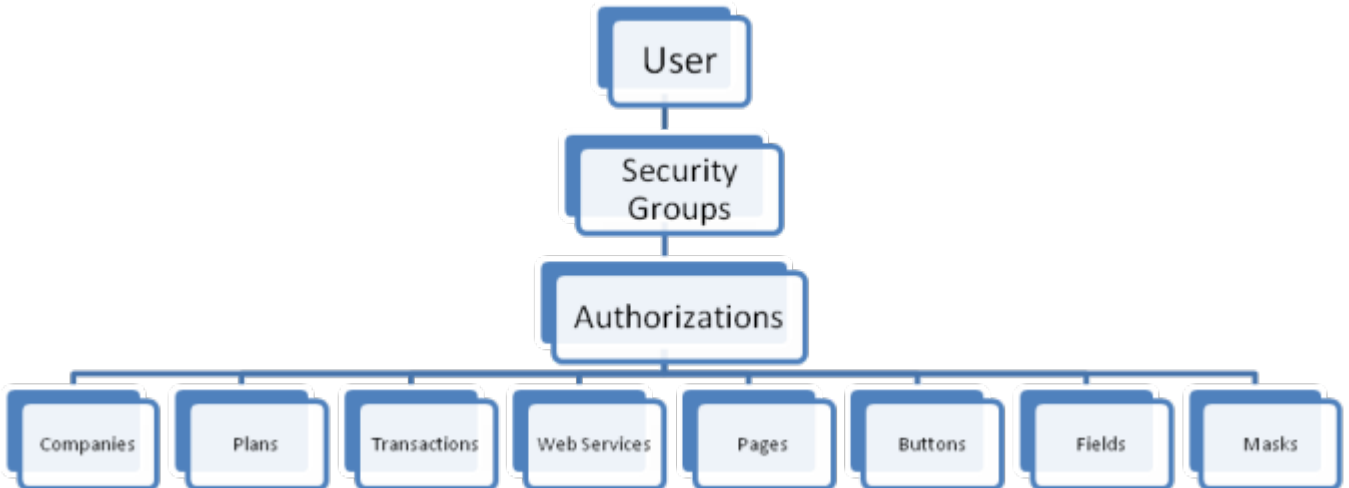
Internationalization and Localization

The OIPA application may be adapted to different languages, regional differences and technical requirements of a particular target market through rules configuration. A configured OIPA-based solution is capable of supporting multiple locales and users of different languages by allowing the co-existence of several languages within the user interface. Configuring a locale and, therefore, adding support for a language though rules, does not require re-engineering or changing system's code.

The system also provides the ability to:

» Translate the content between languages

» Store and display content in multiple languages

» Use regional formats for dates, numbers, and calendars to enter dates

» Display and enter names and addresses in forms native for supported locales

» Handle multiple currencies

» Store country and jurisdiction information that could be used for tax and other purposes

» Allow further customization through configurable extensions to satisfy client-specific localization requirements

The most important internationalization and localization features of the OIPA application are:

» OIPA localization is based on the locale of the current user. A number of locales exist in a con-figured OIPA system and determine which language, translations, date and number format, and so on, will be used when displaying the user interface for a user.

» Data in OIPA are represented in Unicode, the industry standard for the consistent encoding, rep-resenting and handling of text data in most languages.

» Data is stored in the database using multi-byte character types.

» Text data displayed on the screen come through a translation layer (with the exception of text entered by users) where the translation is performed based on the locale of the user. The trans-lation is applied to configured text data, validation and error messages, field labels, and so on.

» OIPA uses Java parsing and formatting facilities to support locale-based formatting of dates and numbers. This "out-of-the-box" formatting can be enhanced or even overridden to satisfy the most diverse customer requirements.

» OIPA has an extensive support for using multiple currencies.

» OIPA allows configuration of the name and address displays to be customized based on the needs of a particular region.

Configuration of locales, translations, screen rules, and so on, is performed by using the Oracle Insurance Rules Palette.

Support for Multiple Currencies Overview

OIPA provides extensive multi-currency support in a single instance of the system, including:

» Currency Entry / Display – the ability to allow users to input monetary values in different currency denominations

» Currency Formatting – the ability for the system to support various formatting and rounding rules based on a currency

» Currency Conversion – the ability for the system to convert money from one currency denomination to another, and track the conversion details

Support for Multiple Currencies

Currency Entry describes the ability for the system to accept entry of monetary values in different currency denominations. What currencies are enabled depends solely on the configuration of the system. The configurable elements include:

» The currency designation for a given field – a field on a screen may be assigned a single currency or a list of acceptable currencies with a default currency.

» The default currency for a product – when no currency is configured for a field, or monetary data is displayed on any screen, the default currency that is configured for the current product will be used to display the currency.

» The default currency for a company – when a screen does not pertain to a particular product, the default currency that is configured for the company the product belongs to will be applied when displaying monetary data.

» The system-wide default currency - when a screen does not pertain to a particular plan or company, the default currency that is configured for the application will be applied when displaying monetary data.

Currency Formatting

OIPA formats monetary amounts that are displayed in the application. There are two completely separate pieces of the functionality: number formatting and currency rounding.

Number Formatting refers to how the number appears to the end user, irrespective of the currency. The way the number appears to the end user is determined by the logged-in user's locale. The locale determines the grouping character, decimal point character, and negative inflection of a number.

Currency Rounding is the process of rounding a number before it is displayed to the user, used in processing, or accepted as input by the user. The rounding rules for a currency are configured in the database as part of the configuration of the OIPA application.

Currency Conversion

Currency conversion is the process of converting a number of units of one currency denomination to another. For example, converting 100,000 USD (U.S. Dollars) to Japanese yen will require a conversion. The conversion that takes place is based on foreign exchange rates. A foreign exchange transaction is the exchange of money from

one currency to another. The foreign exchange rate is a price; the number of units of one nation's currency that must be surrendered in order to acquire one unit of another nation's currency.

The currency conversion in OIPA happens in two places: money movement and configured math. The condition of moving money from one currency to another is automatically detected during transaction processing, so the currency conversion happens automatically. Whenever a currency is converted, there is a cost incurred in converting the money. This currency conversion cost is captured by the system when the conversion takes place.

Configured math supports the ability to convert from one currency to another as a simple formula, and there is no tracking of the currency conversion details that take place in math. When the currency conversion happens, the application will look up the exchange rates for the currencies using the latest exchange rates.

OIPA stores information on currencies, exchange rates, and rules that different market makers use to convert currencies.

Scalability

Scalability refers to the ability of a system to cope with growing loads with stable performance by replicating the system's hardware and software components.

As the following diagram illustrates, the OIPA architecture addresses the scalability of the system by allowing the system resources to be scaled up as needed and minimizing contention on the resources that are most likely to become bottlenecks.
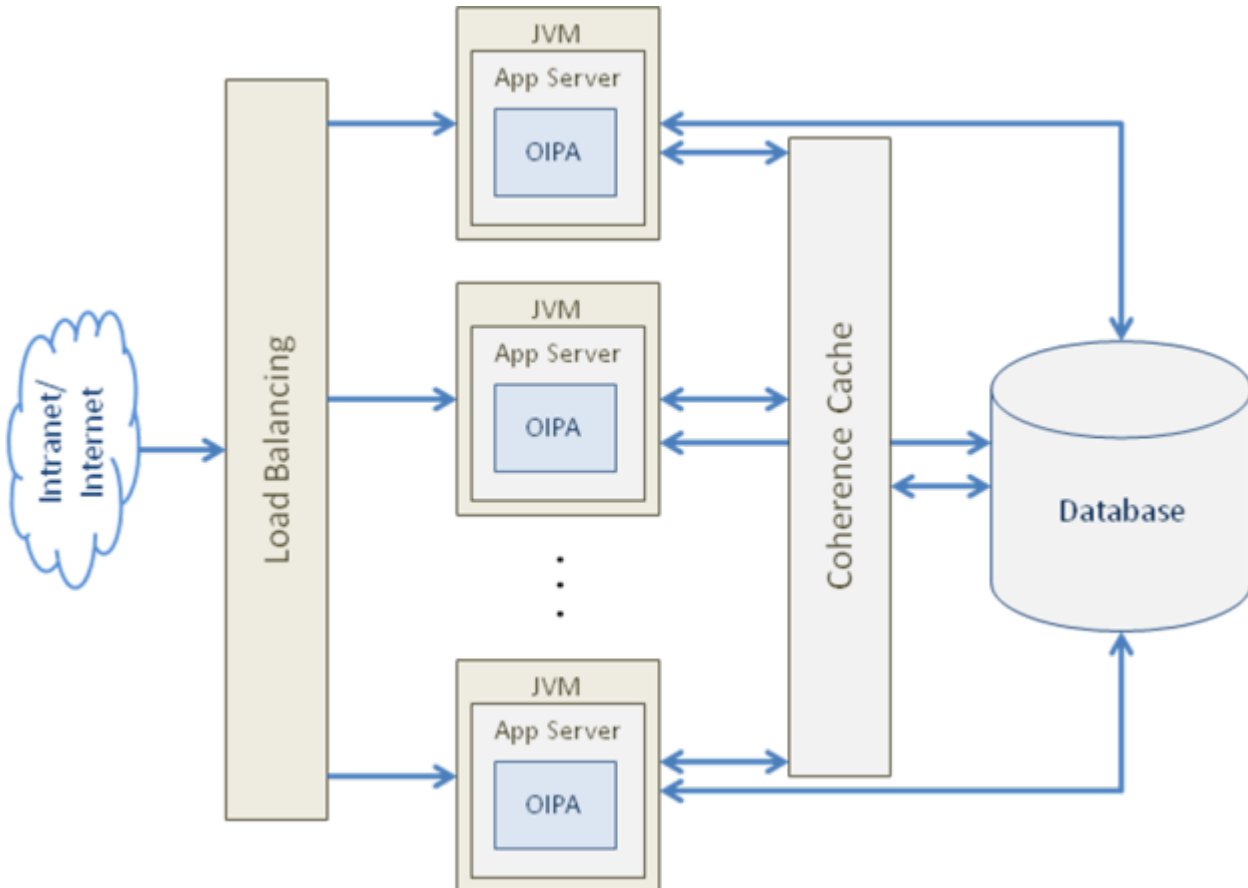

» The system performance may be increased by replicating the JVM/Application server instances that serve the user requests. This requires clustering the servers and imposes an additional over- head of the load balancing.

» It is fairly easy to replicate the application instances, but at some point the database will become a bottleneck. While it is possible to increase the database performance by upgrading the hardware, OIPA tries to increase database performance by optimizing the traffic between the application and the database as much as possible. The generated SQL, returned data sets, data- bases indexes are analyzed and optimized throughout the design, development, testing and con- figuration of the OIPA system.

» Much of the persistent data that rarely change are stored in the distributed Coherence cache that further decreases the load on the database and increases the scalability of the system.

The same scalability approach is applicable to the batch processing performed by the OIPA Cycle components described in the following section.

Cycle

The Cycle subsystem drives processing of transactions through a Cycle Grid, which is comprised of a set of Cycle agents. Each Cycle Agent is a separate JVM instance, which acts as a processing node in the Grid. A special application called a Cycle client submits tasks to the Cycle Grid to direct what type of Cycle processing should be performed. The following is a high-level diagram showing how the different collaborators work together to start processing a Cycle run.
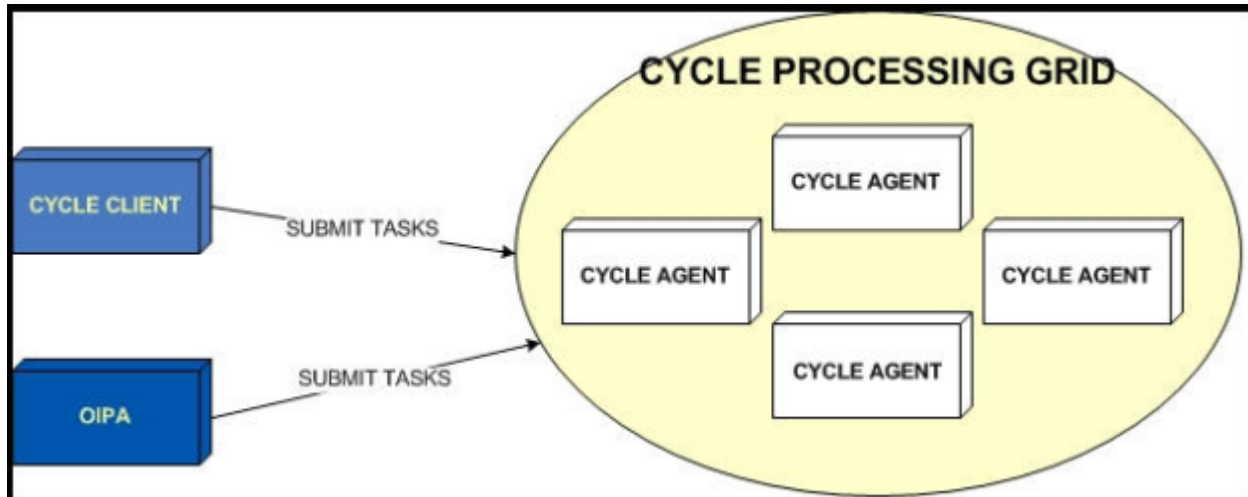
Cycle Agents are the programs that execute the tasks that comprise the Cycle batch process. Each Cycle Agent runs in its own Java Virtual Machine (JVM). Depending on the number of tasks that must be executed during Cycle processing, any number of JVMs may be used to run the desired number of Cycle Agents. The JVMs may be started on one or more physical machines. If more than one machine is used, they will be effectively clustered via the Coherence clustered caching and messaging system.

Data Intake

The File Receiving System is responsible for parsing the Data Intake file and sending appropriate messages to the JMS queue. A listener in OIPA listens for messages on the queue, and when it receives the process operation message (see Data Intake document on OTN for messaging details), it creates a file resumable task to monitor the status of the file. OIPA also has a record resumable task that controls the processing of the individual records in the file. On receiving the process operation message, the record resumable task creates runnable tasks to process the data intake records. The runnable tasks run on all the nodes of the cycle grid thus distributing the workload for speed of execution. The tasks are managed by Coherence. The file resumable task periodically checks to see the status of record processing and updates the file processing status based on it. Once all the records have been processed, the file is marked as complete.

OIPA Bridge/ListenerSubmit TaskJMS queue File Receiving SystemCycle AgentCycle AgentCycle AgentCycle Agent

Caching

Caching is a technique to improve performance by transparently storing data so that future requests for the data can be served faster from the cache. OIPA uses a cache to save configuration data and data that rarely change. Transactions, business rules, currencies and authentication data are just a few examples of the data cached in the system.

Coherence is used as a cache provider. Coherence provides distributed caching services for all nodes in the same cluster. It also provides local caching services within a node.
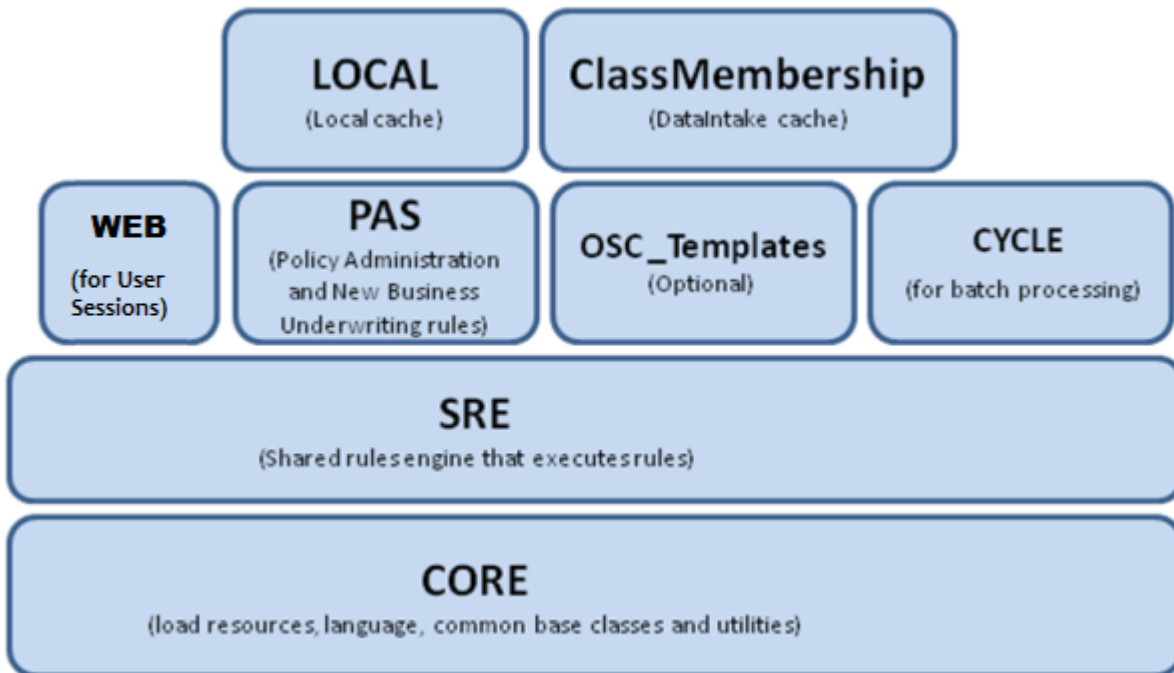
OIPA caches static configuration data and other data that rarely change. There are seven cache regions in the system, each for a different module. Every cache region is configured separately, allowing each one to be configured differently.

The seven cache regions are:

1. Region CORE for shared libraries.
2. Region SRE for shared rules engine components.
3. Region WEB for User Sessions.
4. Region PAS for OIPA specific data.
5. Region LOCAL for providing caching services on the local node.
6. Region OSC_Templates for OSC specific caching.
7. Region CYCLE for batch processing.
8. Region ClassMembership for Data Intake processing.

The Region OSC_Templates for OSC specific caching. This only needs to be configured if OSC is being used, See document "OSC Installation Instructions" for further details. The regions CORE, SRE, PAS, OSC_Templates, CYCLE and ClassMembership are distributed across all nodes in a cluster. The LOCAL region is not distributed. Each JVM has its own LOCAL cache that is specific to it.

OIPA-Based Solution

The following diagram is a high-level deployment view of an insurance administration solution based on the OIPA application:
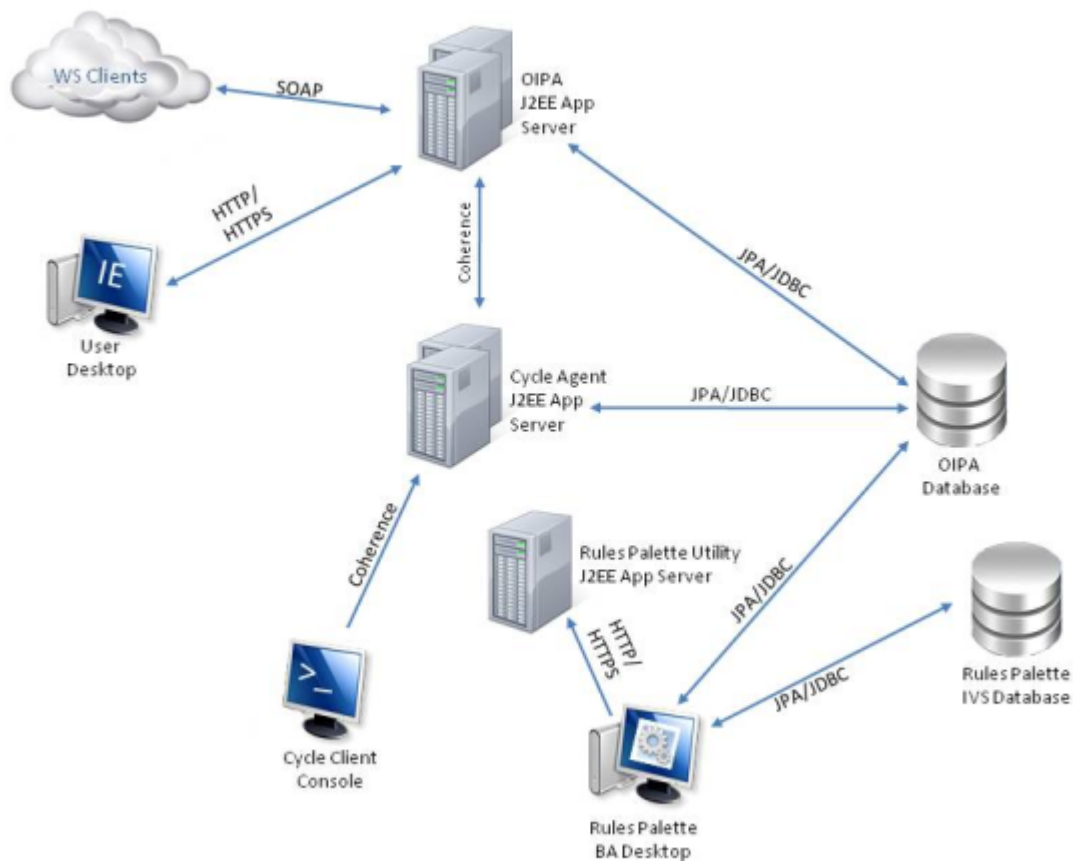

The view contains the following elements:

» **OIPA J2EE Application Server** – This node represents installations of a J2EE Application Server, such as Oracle WebLogic or IBM WebSphere, running instances of the OIPA web application. Every OIPA-based solution will be different depending on the customer's environment, number of users, availability and other requirements. Typically, there will be multiple application servers spread out across multiple machines in order to enable load balancing and failover.

» **Cycle Agent J2EE Application Server** – This node represents J2EE Application Servers running OIPA Cycle Agents. The Cycle Agent adds grid computing to the OIPA-based system, and is responsible for executing batch processing of queued insurance activities. Typically, there will be multiple OIPA Cycle Application server instances per customer installation in order to support high-volume transaction processing requirements.

» **Rules Palette Utility J2EE Application Server** – This node is an instance of the utility application used for configuration and authentication of the Rules Palette instances on the business analyst's desktops.

» **Web Service Clients** – A node that represents external applications or middleware that consume web services provided by the OIPA application.

» **User Desktop** – This represents interactive users that access the OIPA Web application through the Internet Explorer browser. A typical OIPA solution is deployed in a home office of an insurance company and allows access to the server-based web application within a secured intranet.

» **Cycle Client Console** – This node is a console-based client application that initiates specific Cycle processing.

» **Rules Palette BA Desktop** – This node represents the business analyst's workstations that run the Rules Palette. The Rules Palette is a Java desktop application built to configure an OIPA-based solution. It provides a rich user interface including drag-and-drop capabilities to work with business rules, transactions, user security, and so on.

» **OIPA Database** – This node represents an OIPA application database that contains both business data and configuration.

» **Rules Palette IVS Database** – This represents a database that stores versioning information of the system's configuration. The versioning data is used by the Rules Palette that implements a configuration management system with versioning and revision control capabilities.

Configuring a Solution
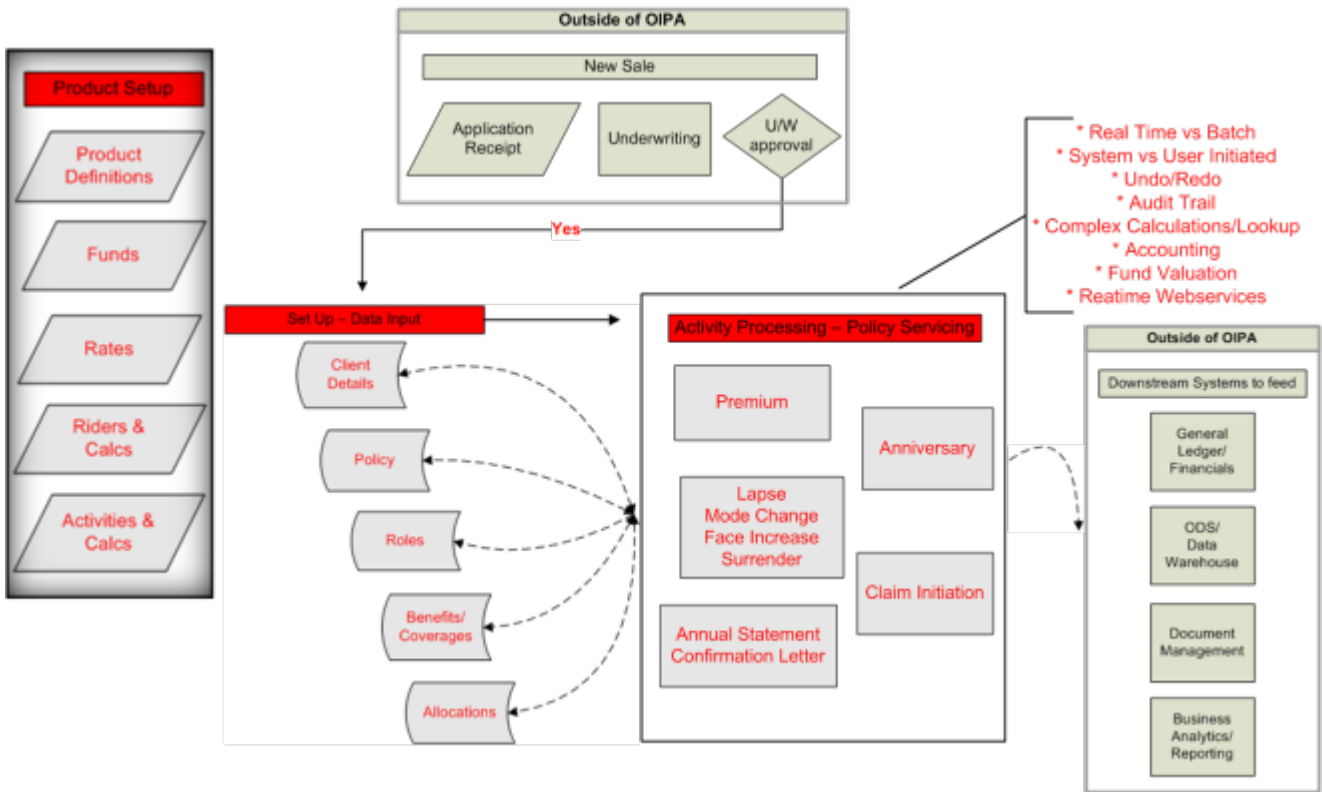
The diagram above represents the processing executed outside of OIPA, as well as the main processing within OIPA using a generic life cycle of an individual life policy.

» **The top block** called **New Sale** represents the New Business and Underwriting processes that execute externally of the OIPA solution. Data from these processes may be transmitted to OIPA through the FileReceived web service or through manual entry via configurable screens.

» On the left, in red, begins the **representation of the OIPA solution processing**, starting with an individual product setup perspective. This represents a portion of the configuration that would constitute a saleable product. After product setup and availability, the data from specific sales of the product create policies in OIPA.

» The diagram highlights a few of the configuration driven elements of a policy; client, policy, role, benefits, coverages, and fund allocation.

» The next block represents the **recordkeeping**, or **Activity processing**. This processing comprises business events or transactions to support Policy Servicing and product required processing (anniversary, premium, annual statement). These sections, all denoted in red, are controlled by rules. All rules are configured (using the Rules Palette) as part of a product implementation.

» The bullet points in red are emphasis on strengths of activity processing within OIPA.

» The data retained in the OIPA database may be sent to downstream systems through real time requests from external sources or through the OIPA cycle process.

Database Optimization

As discussed in the Scalability section, the database performance is paramount to maintaining stable performance of the entire system. Because most of the application's business logic is contained in the configurable business rules, the final database optimization cannot be performed until the configuration step has been completed. Additionally, the performance of the database indexes may need to be fine-tuned with the actual customer's data. This also can be done only after the configuration is finished.

Out of the box, the OIPA database comes with indexes built and optimized for SQL queries that are independent of the configuration. The transactions and business rules configured to implement a customer's business requirements usually contain a significant number of SQL queries as well. Also, the configuration, for example, for search screens, may change SQL queries issued from the application code.

The customer-specific and configuration-dependent queries need to be analyzed to ensure they are efficient, executed quickly and does not retrieve data that are not used. A new set of database indexes may be needed to provide the best possible performance for the customer-specific queries.

The final step of the database optimization should be performed with the actual business data during the performance and load testing of the configured OIPA-based solution.

Extensions and Integration

The ability to integrate the OIPA application with external systems and extend it beyond what could be done through business rule configuration gives the system unlimited flexibility and allows implementation of specific customer requirements. The OIPA system provides several mechanisms for extensibility.

Currently, all extensions are implemented as Java classes that are injected into specific points or levels in the OIPA infrastructure. Extension developers only need to implement the requisite Java interfaces in order to access this powerful OIPA feature.

Primarily, there are three types of extensions: **Anonymous**, **Feature Specific** and **Named**.

1. The **Anonymous** extensions allow for invocation of custom logic using pre-defined methodologies.
2. The **Feature Specific** extensions allow for extending specific OIPA features.
3. The **Named** extensions allow for custom logic at pre-defined points through the lifecycle of specific system events. Named extensions allow for fine-tuned customization.

Anonymous Extensions

Anonymous extensions are provided with data from running transactions, making them powerful tools for integration. Anonymous extensions can execute when an activity is run or when the FileReceived web service is invoked. In the example below, the policy lifecycle includes the OIPA transactions Premium Receipt, Issue and Billing. The first two transactions illustrate how the system can perform messaging over an enterprise service bus (ESB). The last transaction, Billing, illustrates MQ series integration.
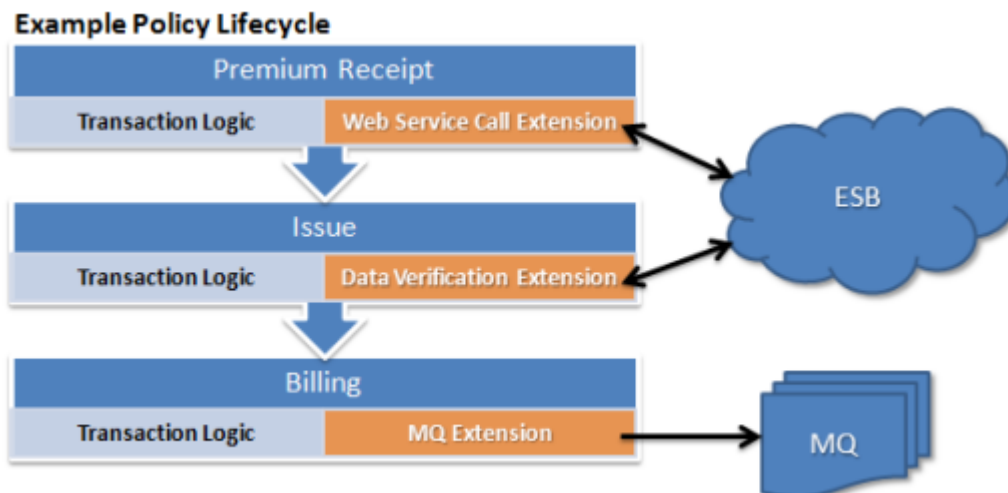
Feature Specific Extensions

Feature Specific extensions provide the ability to enhance a specific OIPA feature. There is only one Feature Specific extension at this time called External Client Integration which can be used to enhance OIPA's client and role management feature to interact with an external agent database.
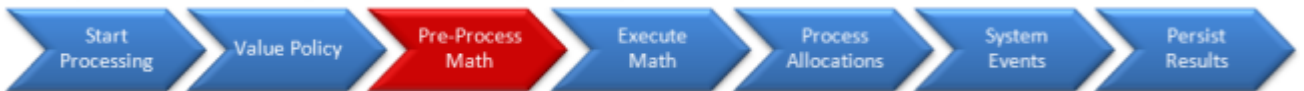
Named Extensions

When fine-tuned control over the application's lifecycle is required, Named extensions can be employed. Named extensions are provided through the Extensibility Framework, which is discussed later in this document.

Below is a simplified rules engine processing example that illustrates how Named extensions can provide pre- or post-processing, or can replace a processing step altogether.
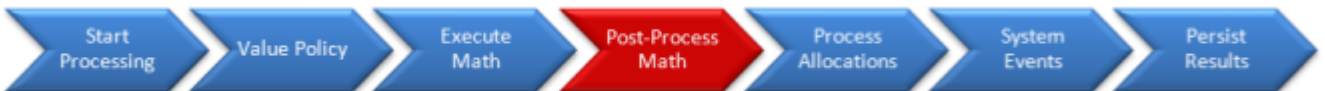
1. Default implementation

| Start Processing | Value Policy | Execute Math | Process Allocations | System Events | Persist Results |

2. Pre-processing added via an extension

| Start Processing | Value Policy | Pre-Process Math | Execute Math | Process Allocations | System Events | Persist Results |

3. Post-processing added via an extension

| Start Processing | Value Policy | Execute Math | Post-Process Math | Process Allocations | System Events | Persist Results |

4. Math processing lifecycle step is replaced with custom math via an extension

| Start Processing | Value Policy | Custom Math | Process Allocations | System Events | Persist Results |

Document Generation

The OIPA application provides users with the ability to generate documents in PDF format. Out of the box, the document generating system uses Crystal Reports. The standard Crystal Reports report files are used as the document templates.

The Rules Palette provides the tools to configure necessary components of the Document Generator. Configuring a document generation involves the following steps:

1. A transaction that generates a document should be configured.
2. A GenerateDocument business rule should be configured and attached to the transaction. The rule prepares business data needed to generate a document and specifies the name of the document template. Each individual transaction that generates documents can be configured to use a specific template.
3. A Crystal Reports document template should be created and made available to the document generation system.

After a transaction that generates documents is processed, generated documents are available on the activity results screen.

The document generation in the OIPA application is implemented as a pre-packaged system extension that internally uses a document generation web service. When implementing a client-specific solution, a different extension can be built and configured to use alternative document generating technologies.

This approach has been employed to implement a document generation extension that uses the OIPA extension framework to integrate with Oracle Documaker. Oracle Documaker is a leading Enterprise Document Automation solution. The OIPA extension allows using Documaker as a document generating technology instead of Crystal Reports.