FUJITSU

WHITE PAPER
FUJITSU INTERSTAGE BUSINESS OPERATIONS PLATFORM - ARCHITECTURE

Fujitsu's Interstage® Business Operations Platform is a next generation business solution and integration platform. Based on the class leading Business Operations Platform from Cordys™, it is designed to truly support the way businesses operate, finally bringing the worlds of business and IT together.

CONTENTS

INTRODUCTION

Fujitsu's Interstage Business Operations Platform (Interstage BOP) is a next generation Business Operations Management solution, designed to truly support the way business operates by bringing the worlds of business and IT together. Interstage BOP is a comprehensive offering which includes business process design, execution, monitoring and improvement. It is based on the Business Operations Platform from Cordys™.

Interstage BOP is designed to help business managers directly align business process implementations with business goals, while facilitating process improvement via unparalleled control and visibility into process metrics and real-time business activity. At the same time, Interstage BOP helps IT managers and developers to rapidly model and integrate their entire enterprise business process landscape, while ensuring that existing IT assets are fully leveraged.

This whitepaper provides an introduction to the architecture of Interstage BOP and explains how it uniquely enables customers to improve their business operations. The intended audience for this whitepaper are solution architects and technical users who wish to obtain a thorough understanding of the technical aspects of Interstage BOP.

The whitepaper delineates the technical capabilities for Interstage BOP. It starts out by describing the platform vision and goals, followed by a high-level overview. Subsequent sections are more technical and address the design-time and run-time architecture, as well as the set of applicable standards.

ARCHITECTURE VISION AND GOALS

We, at Fujitsu, see it as our mission to help our customers improve their business operations with world-class, process-oriented software which allows them to change and innovate the way they do business with greater speed and flexibility. This mission is translated in the following set of architecture goals:

- **Integrated platform** –
  Giving simplified installation and maintenance, thus reducing total cost of ownership.
- **Browser-based access for all users, including administrators and application designers** –
  Enables any user, inside and outside the company, to work with Interstage BOP system using just a browser
- **Application development for technical and non-technical users** –
  Bridges the gap between business and IT by enabling non-technical users to participate in the development process
- **Standards compliance** –
  Enables easy integration, thus reducing total cost of ownership
- **Extensible environment** –
  Easy extensibility drives down the total cost of ownership
- **Internet and intranet deployability** –
  The same platform can be used for cloud computing and on-premise
- **Linear scalability** –
  Linear scalability on commodity hardware keeps total cost of ownership low
- **High availability** –
  Business critical systems demand high availability
- **Multitenancy** –
  Cloud computing is a core feature of the platform

OVERVIEW OF INTERSTAGE BUSINESS OPERATIONS PLATFORM

Interstage BOP is inherently integrated, as it is built as a single product. All features are based on a single technology.

MODEL DRIVEN DEVELOPMENT

A key objective of Interstage BOP is to enable business users to participate in a model-driven application development. This has been done before: many platforms allow business people to participate through contribution of models which programmers then take as input. However Interstage BOP takes a radically different approach: the model is the application, not merely an input to a programmer. To enable that, Interstage BOP application development is predominately model-driven.

The Interstage BOP modeling environment is built as an application on top of the Interstage BOP runtime environment. It has a number of significant features that make it:

- **Scalable, robust and secure**:
  All platform runtime features like scalability, high availability and security directly contribute to the design-time environment.
- **Testable**:
  There is no need to install another product to enable testing.
- **Available everywhere**:
  Every Interstage BOP installation comes with built-in design capability.

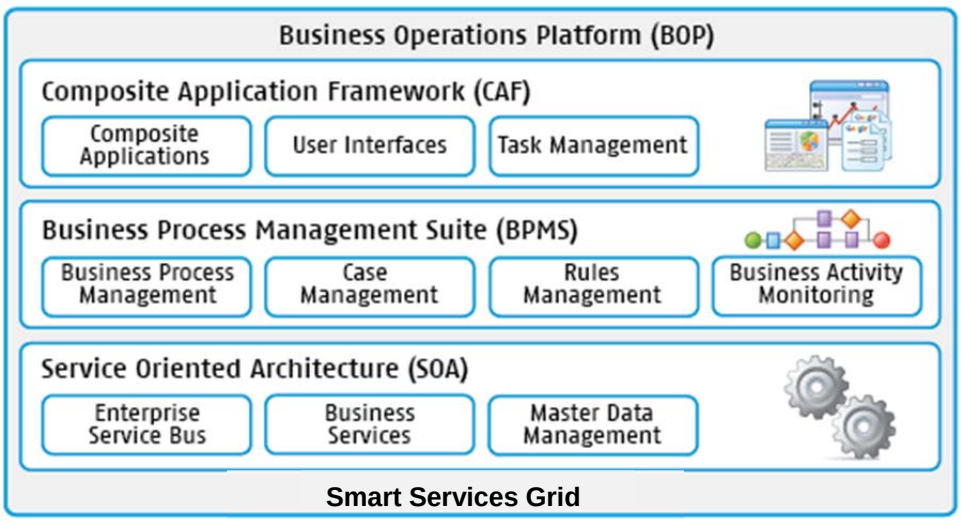
The full functionality of Interstage BOP is available through a variety of completely browser based user interfaces. Be it system administration, modeling of applications or an end user application such as claims handling, all interactions are browser based. This makes it possible to deploy Interstage BOP, as well as Interstage BOP based applications, in both Internet and Intranet scenarios. It also enables quick and hassle-free involvement of new users.

A new participant in a project needs only a web browser; there is no need to install anything locally. Interstage BOP provides support for the industry's most popular browsers, Chrome, Firefox, Internet Explorer and Safari.

As mentioned above, the design-time environment is based on the platform runtime environment. This however does not imply that the application needs to run in the environment where it has been designed. Standard development practice is to employ a DTAP setup (Development, Testing, Acceptance and Production). Different environments are used for different phases of the software development cycle. The platform has provisions to package and deploy applications to facilitate this approach.

Interstage BOP provides the following basic features:

- **High availability**:
  Mission critical applications must be continuously available. To ensure high availability, Interstage BOP can be deployed on a network of systems, ensuring there is no single point of failure.
- **Scalability**:
  Enterprises handle thousands of business processes in a day. Interstage BOP is built for this task. It can scale vertically (scale up) as well as horizontally (scale out). Horizontal scalability is accomplished with commodity hardware.
- **Multitenancy**:
  Cloud computing scenarios demand multiple organizations, called tenants, to share the same infrastructure. Multitenancy is a basic feature of Interstage BOP that can also be useful in some on-premise scenarios.
- **Security**:
  Interstage BOP has an advanced set of security measures, including access control lists, auditing, encryption and sandboxing.
- **Service orientation**:
  Service-oriented architecture is the predominant design principle in modern enterprise systems. Service orientation belongs to the very core of the platform. All interactions are done through services.



Interstage Business Operations Platform

DESIGN-TIME ARCHITECTURE

This chapter drills a little deeper into the design-time architecture of Interstage BOP. After a quick introduction to our model-driven approach, it describes the highlights of the integrated metamodel, the approach to team development and the anatomy of a modeler. The closing section provides an overview of the standard facilities of Collaborative Workspace (CWS).

MODEL-DRIVEN

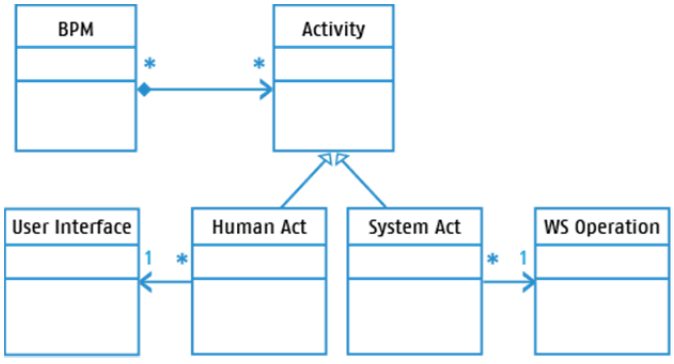
Interstage BOP takes a model driven approach towards application development. The key principle is 'What you model is what you execute'. All modeling activities are done in the **Collaborative Workspace** (CWS), a browser-based integrated modeling environment allowing definition of all types of models: business process, user interface, application object, WSDL, etc. Model developers use a browser to create new models or optimize existing ones. Most models are represented through graphical models, featuring a responsive rich user interface.

INTEGRATED METAMODEL

The models are all based on a single integrated metamodel. This model not only captures the structure, but also information on the (graphical) visualization, constraints, the building and packaging procedure, etc. A key objective of CWS is to ensure consistency; any application that passes the build step should be deployable. This is important during refactoring. Renaming a model should not cause an inconsistency where the 'referrer' holds on to the old name while the 'referee' has a new one.

Each model has an associated Java class that takes care of the runtime behavior inside the CWS service. The XML document representing the models is stored in a relational database through XDS.

A simplified view of a part of the metamodel is shown below.



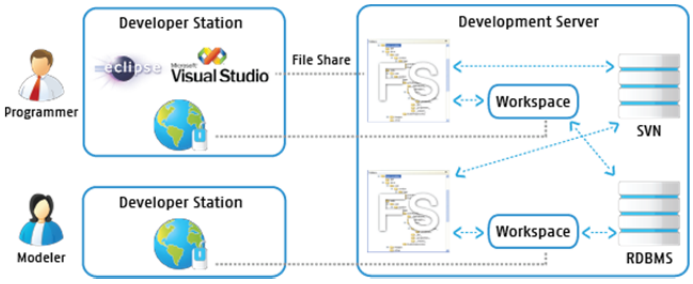
Interstage BOP Integrated Metamodel

TEAM DEVELOPMENT SCENARIO

A business application of reasonable size is usually a team effort. Development teams normally use a **Software Configuration Management** (SCM) product to keep track of revisions of the models and code. CWS has been designed in such a way that SCM products can be plugged into it. The plug-in is available out-of-the-box that supports Subversion.

The SCM interaction is implemented through the File Synchronizer. This facility synchronizes the file system directory and the CWS. The standard SCM features like check-in and check-out are available to all users, as part of the CWS browser interface.

Some team members might be programmers, using Eclipse or Microsoft Visual Studio. Such file-based Integrated Development Environments (IDEs) are also supported through the file synchronizer of CWS.



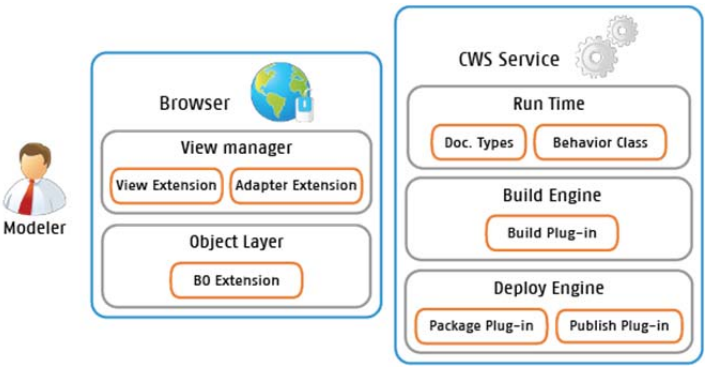
Interstage BOP Team Development

ANATOMY OF A MODELER

Interstage BOP Business Operations Platform provides several out-of-the-box modelers for e.g. Business Processes, User Interfaces, Organization Models, etc. All these modelers are built on an extensible and completely meta-model driven designer platform called CWS. In this section, we will look at the anatomy of a modeler, and explain how the modeler is built.

The first step is to work on the 'class' and 'view' definitions of the metamodel. The view definition defines the user interface for this model. Both definitions are expressed in XML. These two definitions form the input to a generator which produces classes like view extension, adapter extension, build plug-in, etc., for use in the front- and back-end of CWS. The modeler developer adds the custom behavior as needed, and then builds the modeler.

The following diagram illustrates how the various classes are used while running CWS.



Interstage BOP CWS Models

STANDARD CWS FACILITIES

The CWS framework provides a set of standard facilities to all modelers:

- **Where used**: Based on the associations between models, as expressed in the metamodel, CWS automatically provides "where used" overviews. This is an important tool during impact analysis and refactoring.
- **User interface**: Common look and feel across modelers is facilitated through an advanced UI framework.
- **File system synchronization**: The content of a CWS workspace can be synchronized to a file system directory, to enable interaction with SCM tools, use of file-based editors and IDEs. It also helps in easy exchange of workspace content.
- **Import/export**: A plug-in framework supports generic import/export of models. XPDL import/export ships out-of-the-box, other import/export plug-ins can be developed as needed.
- **Build:** The build engine takes care of building all the changed documents. The extent of what is to be built is determined based on the associations between the models. This prevents rebuilding unchanged models. The actual build behavior is specifically implemented for each model.
- **Package:** The packaging framework takes the build output of all models of a project to create the deployable Interstage BOP Application Package (CAP).
- **Publish**: For testing purposes, models can directly be published to the development system. This is taken care of by publishing the framework. This framework also uses the associations to determine what needs to be published.
- **Tagging:** To enable easy lookup of models, it is possible to attach user defined tags to them. This is a standard feature of CWS.

RUNTIME ARCHITECTURE

At runtime, Interstage BOP is comprised of a set of web service containers, connected through a Smart Services Grid. In this chapter we will first look at the logical view, followed by the deployment view, an explanation of Interstage BOP multitenancy and an overview of the runtime services.

LOGICAL VIEW

The logical view of Interstage BOP architecture shown below has a strong resemblance with the diagram showing the Overview of Interstage BOP. The following diagram highlights the technical aspects of Interstage BOP.

There are three main layers:

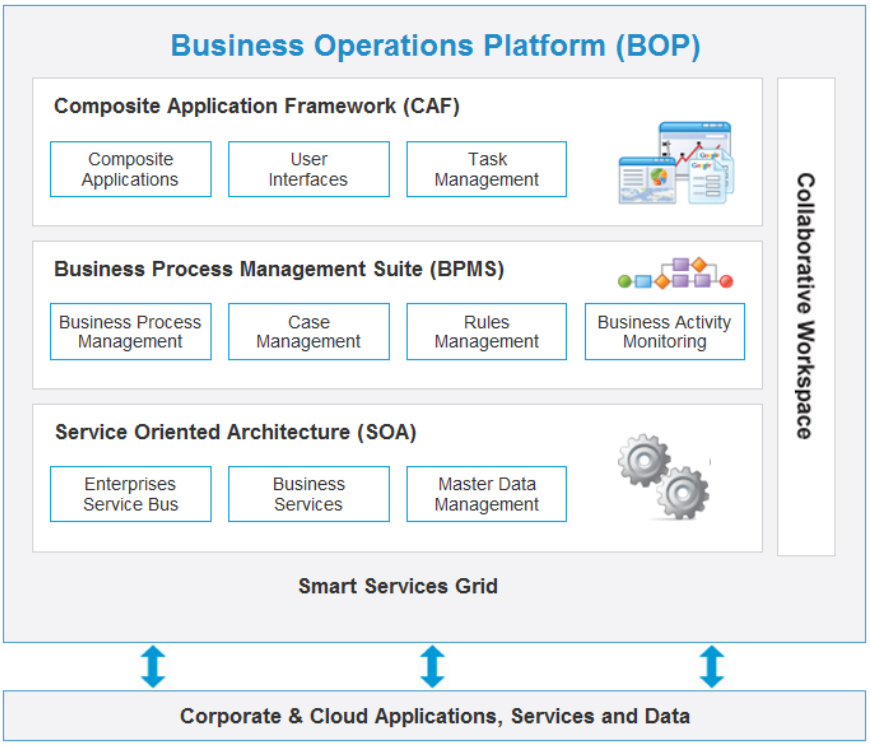
- **User Interface Layer**
  This layer contains all the User Interface components like dashboards, the Inbox and the application user interfaces developed through Interstage BOP. These user interfaces are built on top of the business services as defined in the next layer.
- **Business Services Layer**
  The business layer hosts services relevant to the business domain; notable examples being Business Process Management (BPM), Case Management and Business Activity Monitoring (BAM). These services are built on top of the SOA layer.
- **Service Oriented Architecture Layer**
  This layer provides the fundamental Smart Services Grid functionality used by the other layers.

Later in this chapter, we will look into the details of each component.

DEPLOYMENT VIEW

All runtime components are linked through Interstage BOP Smart Services Grid. The Smart Services Grid provides three main facilities:

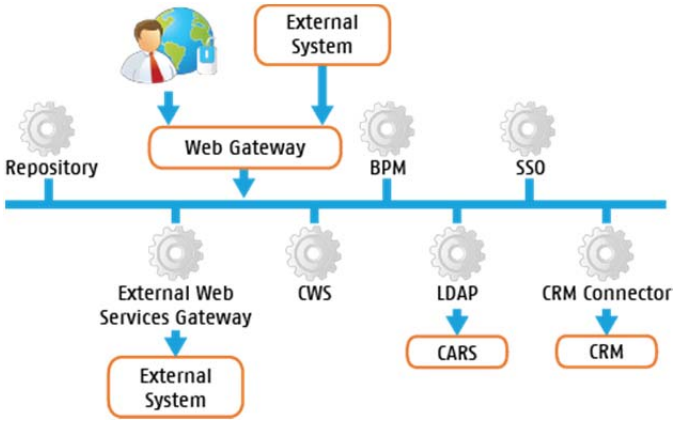
- **Routing of SOAP messages**
  The SOA is entirely SOAP-based. The services deliver their XML messages to the Enterprise Service Bus (ESB). Based on the service registry – stored in an LDAP7 directory called CARS – it knows the details of all the services. Given the required quality of service, it chooses a channel and delivers the message to the recipient. The messages can be transferred over a variety of protocols, ranging from plain TCP/IP sockets to message queues.
- **Load balancing**
  As the load increases, there comes a point when not everything can be handled on a single system; multiple systems can be used to run the same service, thus sharing the load. The ESB has pluggable, load balancing algorithms that decide which service instance to address.
- **Fail over**
  In case one of the service instances fails, load should immediately be moved to other instances and business should go on as usual. This is taken care of by the fail over features of the ESB. Details of these features are discussed in the ESB section.



Logical view of the Interstage BOP architecture

BUS VIEW

The following diagram shows a 'bus' view of Interstage BOP. All participants on the 'bus' are equal, so there is no central 'bus coordinator' or 'single point of failure'. The Web Gateway is depicted with a different icon because of its special role as a 'bus client'. It does not have a role in requests between the peers. The service icons in the diagram represent a random set of the service groups in a standard Interstage BOP system. A service group denotes a conceptual service.



Interstage BOP Bus view

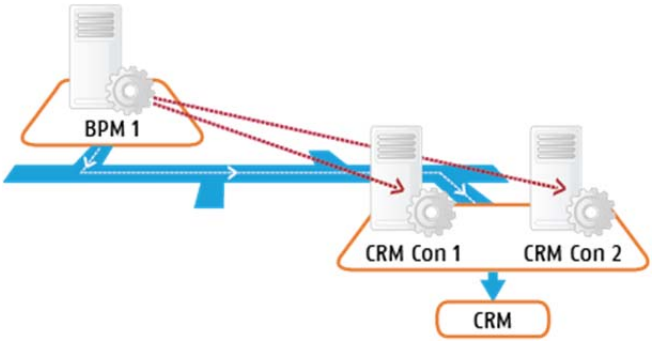
The actual implementation is through a 'service container'. To provide load balancing and fast fail over, one service group might be implemented through multiple service containers, most of the time running on different systems.

Following is a brief description of some of the services:

- **Web gateway**: This actually is not a service on the bus but just a client. It provides an access point for users and external systems using Interstage BOP web services.
- **External web services gateway**: This service, often called the UDDI connector, links external web services to the bus.
- **LDAP**: Interstage BOP uses an LDAP directory, called CARS, as a runtime repository for certain types of information. This repository is accessed through the LDAP service.
- **CRM connector**: The Customer Relationship Management (CRM) connector is an example of a connector to any Enterprise Information System (EIS) or legacy system. Interstage BOP allows developing connectors specific for the various enterprise information systems in an organization. This connector bridges the proprietary protocol of the EIS with the standard protocol used on the bus.

The following diagram clarifies the above explanation by depicting the interaction between an example deployment configuration of the BPM engine and a CRM connector. The BPM engine is deployed as a single service container on one system, whereas, the CRM connector is deployed as two service containers, each one running on its own system.

The BPM engine directly connects to one of the two CRM connectors to submit its request.



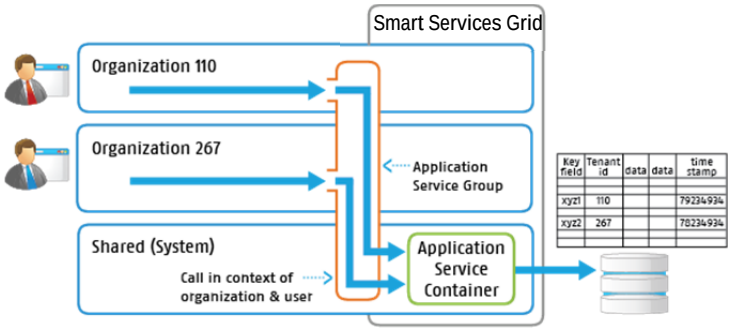
Invoking a web service

MULTITENANCY

The Smart Services Grid used within Interstage BOP represents the physical or deployment aspect of the Interstage BOP architecture. The organization concept represents a logical concept in Interstage BOP. All functionality is invoked in the context of a user and the organization of that user.

The organization is not located on a specific machine. All service containers, wherever located, can execute the functionality on behalf of a user in the context of that organization. So, if a user starts a user interface, it will be in the context of an organization of which the user is a member. And if the user interface invokes a call on a service container, it will execute in the context of that organization and user.

Before it is executed, the role of the user is validated against the **Access Control List** of the service. If the user does not have the correct credentials, the logic will not be executed. A single user can exist in multiple organizations and have different roles in those organizations.



Invoking a web service in a multitenant environment

Service containers exist in the context of an organization. The functionality in a service container is exposed via a SOAP or REST interface. When a SOAP call is initiated, it is done so in the context of an organization, the Smart Services Grid then takes care of finding the service container in the organization that implements this interface and then executes it. For efficiency and reuse, a common shared organization called System acts as a 'fall back' mechanism for other organizations.

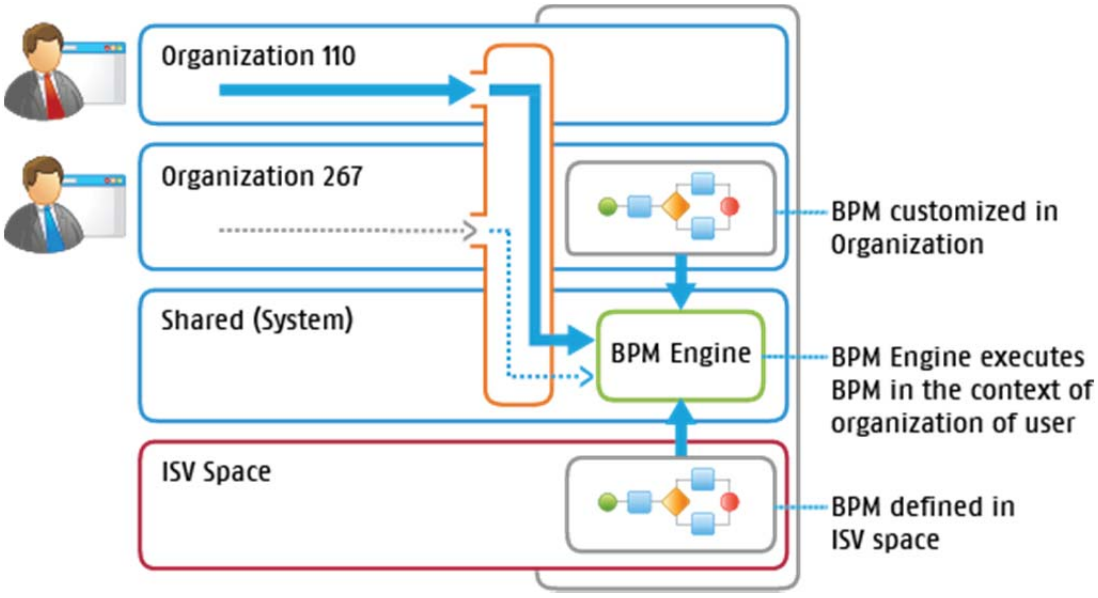
If the SOAP call is not implemented in the current organization, the call is delegated to the shared (system) organization and, if found, is executed on a service container in that organization. But that is done in the context of the user and the original organization.

All Interstage BOP service containers are organization-aware and use multitenant data stores to persist their data. When an Independent Software Vendor (ISV) builds a new application, it is the responsibility of that ISV to take care that the application data store (e.g. a database) supports multitenancy. The service container of the application can be deployed into the System organization to allow all organizations to use this new application.

The content of the application (e.g. the business processes and user interfaces) can be stored in the context of the organization (or tenant) and is only available within the context of that organization. This situation is typical when developing or customizing an application. Development starts in the context of the organization owned by the ISVs.

This is shown in the diagram below. Although the ISV space is drawn in the same fashion as an organization, it is, technically speaking, not an organization. But content (e.g. application definitions such as user interfaces and business processes) is always stored in the context of an organization or the ISV space. Wherever a service container loads its content from, the context is always based on the organization of the user on whose behalf the logic is executed.

So, taking the customization of an existing application as an example where a business process is modified at an organization level, as shown in the following diagram. The user of "organization 110" will be using the business process defined in ISV space but the business process will run in the context of "organization 110". A user in the "organization 267" will use a customized version of this business process stored in its own organization and this business process will run in the context of "organization 267".



Tenant-customized business process model

USER INTERFACE LAYER

This section provides further details on the 3 layers (User Interface Layer, Business Services Layer, and Service Oriented Architecture Layer) and each of the components depicted in the Logical view.

USER INTERFACES

INTERSTAGE BOP USER START PAGE (CUSP)

Interstage BOP User Start Page (CUSP) is a generic and task-driven view of the product functionality presented to the user. The CUSP view shows a non-hierarchical navigational model with the roles abstracted from the view.

The key elements of Interstage BOP User Start Page are:

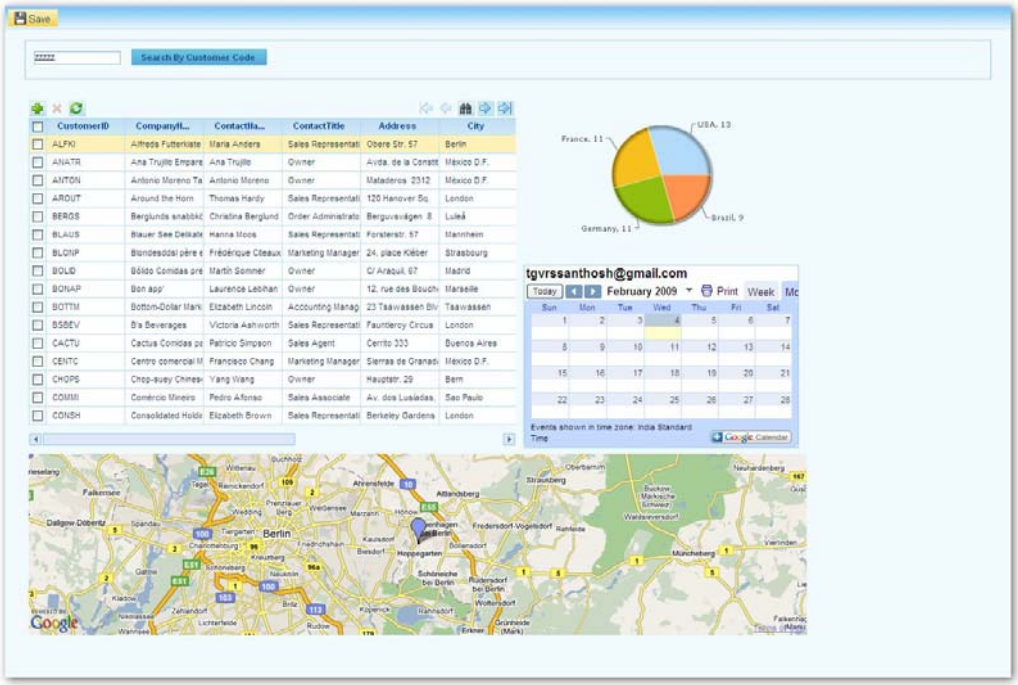
- A **Most Used Tasks** list that consists of the tasks and shortcuts available to a particular user, based on the roles and permissions assigned, and the usage pattern of the user.
  - This will be shown as a list of tasks.
  - The most frequently used tasks will move up the list with progressive usage.
  - The most frequently used tasks come with default categories, and are able to be 'Live Categorized' by the user.
  - Typical examples of a task would be user management, create a user interface, create a business process, etc.
- A **Most Recently Used** list that summarizes the last and latest used artifacts in the system.
  - This list will be specific to a user.
  - The initial view will become empty and the list of most recently used tasks will appear only as the system artifacts are used.
  - The system artifacts listed here come with their default categories, and are able to be 'Live Categorized' by the user.
  - Typical artifacts can be the files, the forms, business process models, etc.
- An Organization Switcher to switch organizations since the tasks are invoked directly and the roles are abstracted.

USER INTERFACE MODELER

Interstage BOP User Interface Modeler makes it easy to design effective user interfaces for composite applications. With the UI modeler, power users can quickly create custom applications that leverage the full power of Interstage BOP, link directly to services running on Interstage BOP ESB, and provide full transactional capability. The UI modeler features a WYSIWYG (What You See Is What You Get) approach and rich UI controls to dramatically streamline and simplify the process of converting ESB functionality into rich, intuitive, process-centric applications that run in a browser environment. The AJAX technology ensures a rich and responsive user experience

The modeled user interfaces in Interstage BOP are based on the XForms standard, as defined by W3C. XForms is an XML format for the specification of a data processing model for XML data and user interface(s), such as web forms. Interstage BOP provides a rich set of UI controls including basic elements like 'check boxes' and 'radio buttons', but also advanced controls like 'AppPalette' and 'Google map'. The environment is extensible through the notion of composite controls. This technology allows building UI controls of any complexity for delivery as reusable UI controls.

The forms are stored in the XML store and delivered through the XForms Service Container. One of the key responsibilities of this service container is to translate the form into the language applicable for the current user.



Rich set of user interface controls

DASHBOARDS

Building a dashboard in Interstage BOP is as simple as building any user interface. Every Key Performance Indicator (KPI) or business measure defined in Interstage BOP Business Activity Monitoring comes with a composite control that provides a chart representation of that business indicator. These charts are placed on a user interface to build a rich and interactive dashboard.

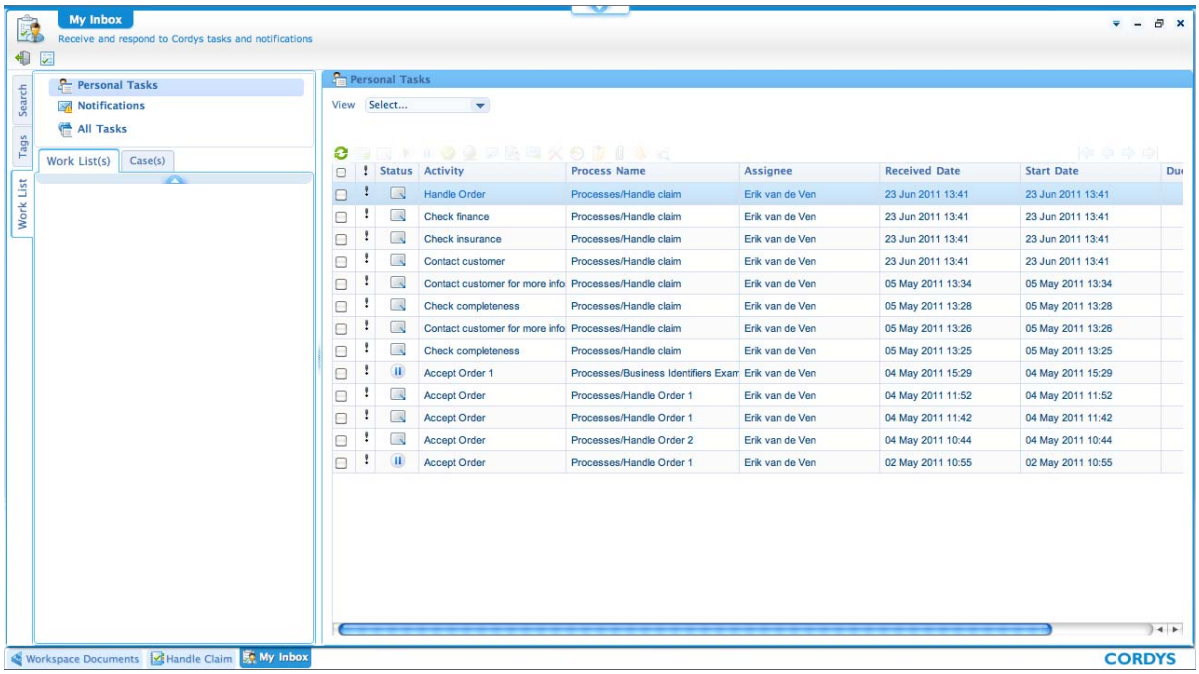
Interstage BOP Dashboard

INBOX

The most common user interface for Interstage BOP users is the Inbox. To facilitate this, an advanced user interface was developed that is fast and highly configurable, to suit all user needs.

- **Case Management integration**:
  - The Interstage BOP Inbox has a seamless integration with the Case Management solution.
  - A Case worker can view all the tasks for a particular case model, grouped by case instance, which he/she has access to work.
  - The Case Task view of the Inbox is a consolidated view of all the active activities for a particular case instance.
  - This can be considered as a completed dashboard, from where the user can plan follow up activities, raise events, attach documents and also perform administrative activities like suspend, resume, forward, delegate, etc.
  - The case worker will be able to view the complete case history from the case Inbox.
- **Customization possibilities in the Inbox**:
  - Inbox has customization possibilities in both 'representation' and 'behavior'.
  - The user can select/change language and date format.
  - The user can customize the list view of the Inbox to show the business data, so that it will be easy for the case worker to select the task that they want to work on.
  - The "view customization" is possible at personal Inbox level and also at the role/team/work list level.
  - Application developers can hook their custom JavaScript code via the customization hooks provided for different events like onBeforeCommit, onBeforeFollowup, etc.,.

The Inbox interacts with the Notification Service in the backend to fetch work items and update statuses



User Inbox

BUSINESS SERVICES LAYER

BUSINESS PROCESS MANAGEMENT

Given our focus on process-oriented software, the BPM engine is one of the most important components in Interstage BOP. Processes are defined as BPMN compliant graphical models in CWS. The build step on such a model results in an XML definition which gets interpreted by the BPM engine at runtime.

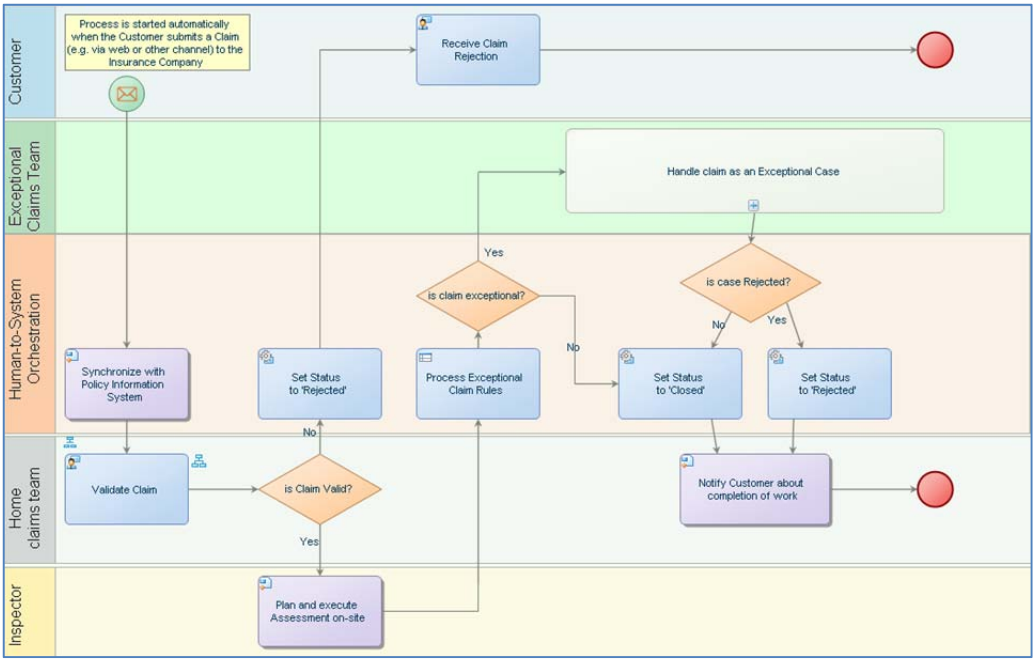
Some highlights:

- **Supports short-lived and long-lived processes**: Short- lived processes do not involve user interaction and do not log progress information. This makes short-lived processes ideal to express fragments of logic. Long-lived processes log their progress in the process instance tables and can involve user interaction.
- **Fast and scalable**: If sufficiently powerful hardware is available, a single BPM engine can execute thousands of short-lived business processes per second. If a single system cannot handle the load, the BPM engine can be scaled out across multiple systems.
- **Part of Business Process Management service group**: In a default Interstage BOP deployment, the BPM engine is part of the 'Business Process Management Service Group', together with Case Management. To optimize performance, this service group runs the following services in an embedded mode:
  - Rule Engine
  - Binary parser
  - WS-AppServer
  - CoBOC
  - Data transformation
- **Process Instance Manager**: Execution of long-lived processes is tracked in the **Process Instance Manager** (PIM) tables. The Process Instance Manager user interface provides access to this data.
- **Multitenant**: Like any Interstage BOP service, the process engine is multitenant enabled, so multiple tenants can have their own process definitions and collections of running processes.
- **WS-AppServer Integration**: High speed transactional processing can be accomplished by embedding WS-AppServer in the BPM service container. This allows short-lived process to directly call WS-AppServer classes in a transactional manner.
- **Crash recovery**: The BPM engine restarts a crashed process from the last recovery point as captured in the Process Instance Manager.
- **Reliable messaging**: When using a reliable message transport, the BPM engine will coordinate the transactions on the process instance manager tables and the message- oriented middleware to ensure that the web service operation is executed once, including the handling of the service response.

A real world example of a BPMN flow in Interstage BOP is shown in the following diagram. The example shows a long-lived insurance claim process across different departments and external parties. It is depicted as a yellow swim lane at the bottom representing the "Inspector" who reviews the incident on-site, per request of the insurance company.

This flow shows clearly that Interstage BOP orchestrates human tasks (via user interfaces) and system tasks (e.g. web services updating the claim status during the claim handling) in a single process model.

Rule Engine and Data transformation are all invoked from this BPMN flow, not necessarily visible for the business end user, but visible for the functional administrator or process analyst.



Business process example

CASE MANAGEMENT

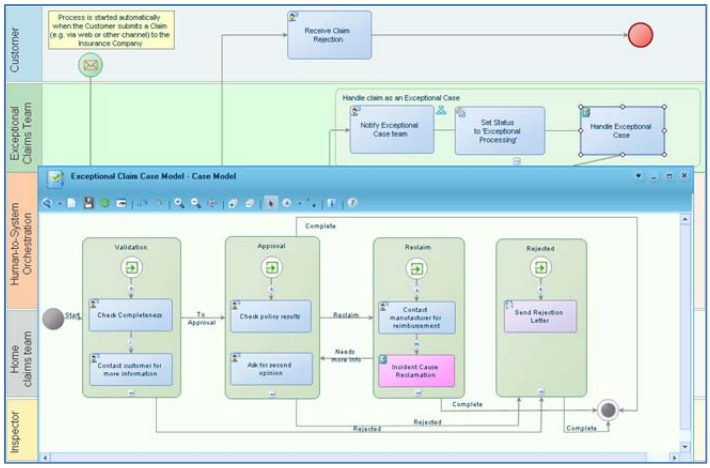
BPMN is suitable for processes that are well-defined and uniform. In day-to-day life, however, processes often vary on a case-to-case basis and the knowledge worker in the process wants to decide on the next steps to be taken. This is the 'sweet spot' for Case Management.

At its core, a case model is a state machine. Users can decide to express their case model as a set of interrelated states, but they can also opt for a simplified model with just one implicit state. The activities of a case model are related to each other through "follow-up relations". The 'build' step of a case model produces XML, which at runtime gets interpreted by the Case Management state machine.

Some highlights:

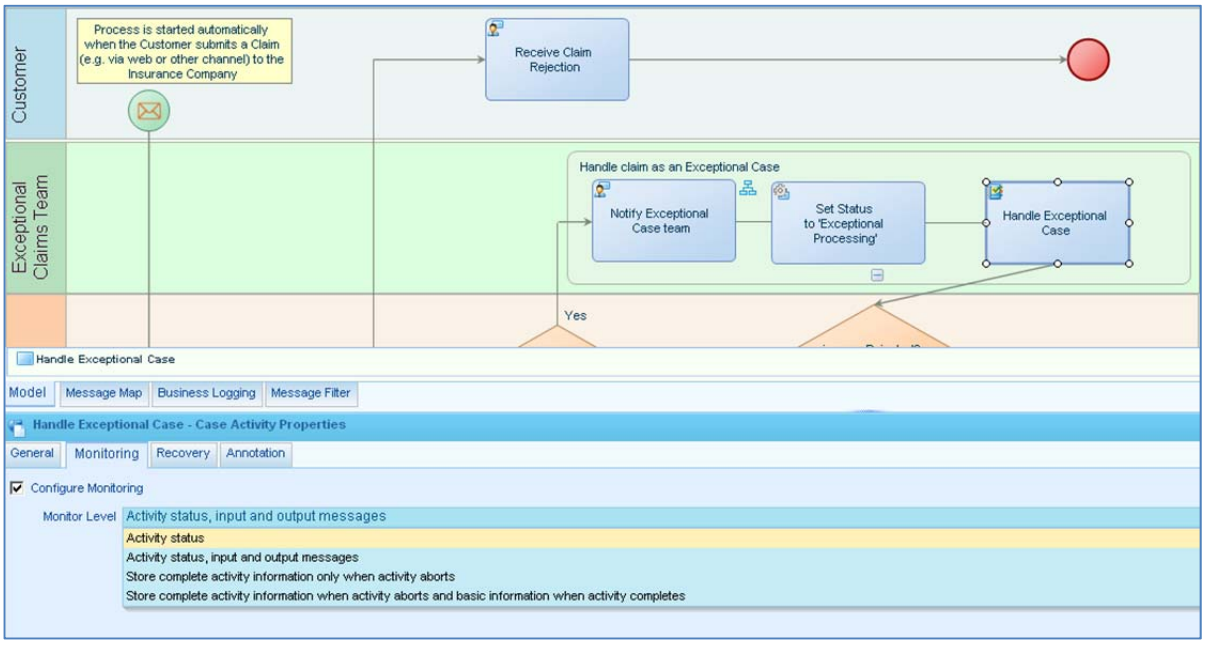
- **Fully state machine based**: A case model is converted into a state model, including aspects like follow-up relations that might not seem state-related at first glance.
- **Standard SCXML format**: The state definition complies with the SCXML10 standard as defined by W3C.
- **Case data management**: Case models define the data structure applicable for that case.
- **Case Instance Manager**: The progress of a particular case, as well as any changes to the case data, is tracked in the case instance tables. The Case Instance Manager provides access to this data.
- **Part of Business Process Management service group**: In a default Interstage BOP deployment, the case engine is part of the 'Business Process Management' service group, together with the BPM engine.

The following diagram shows a screenshot of the case model invoked from the Main BPM flow which was shown in the previous diagram. It demonstrates how the BPM flows and case models can invoke each other. This case model has four functional states and the arrows depict the state transitions which are often event- and trigger-based.



Case Model

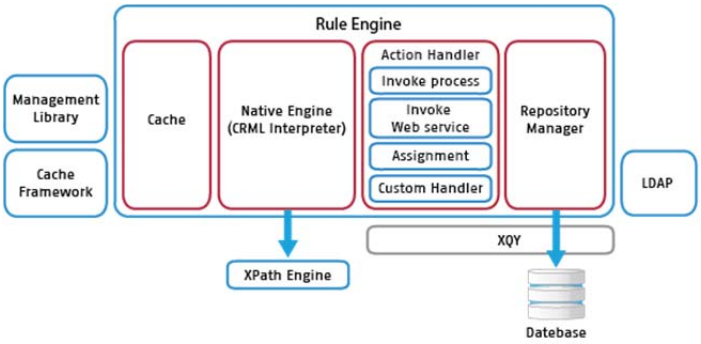
Comparing the Main BPMN flow given in the previous diagram with the one given below, you can see that Interstage BOP has the option 'to collapse and expand groups of activities.' The embedded subprocess labeled 'Handle claim as an Exceptional Case' is expanded by the process designer in the following diagram into detailed activities. The last one of which is labeled 'Handle Exceptional Case' invokes the case model. The invocation can be configured via the properties boxes allowing process designers to tune runtime behavior exactly to their needs.



Expanded view of subprocess 'Handle claim as an Exceptional case'

RULES MANAGEMENT

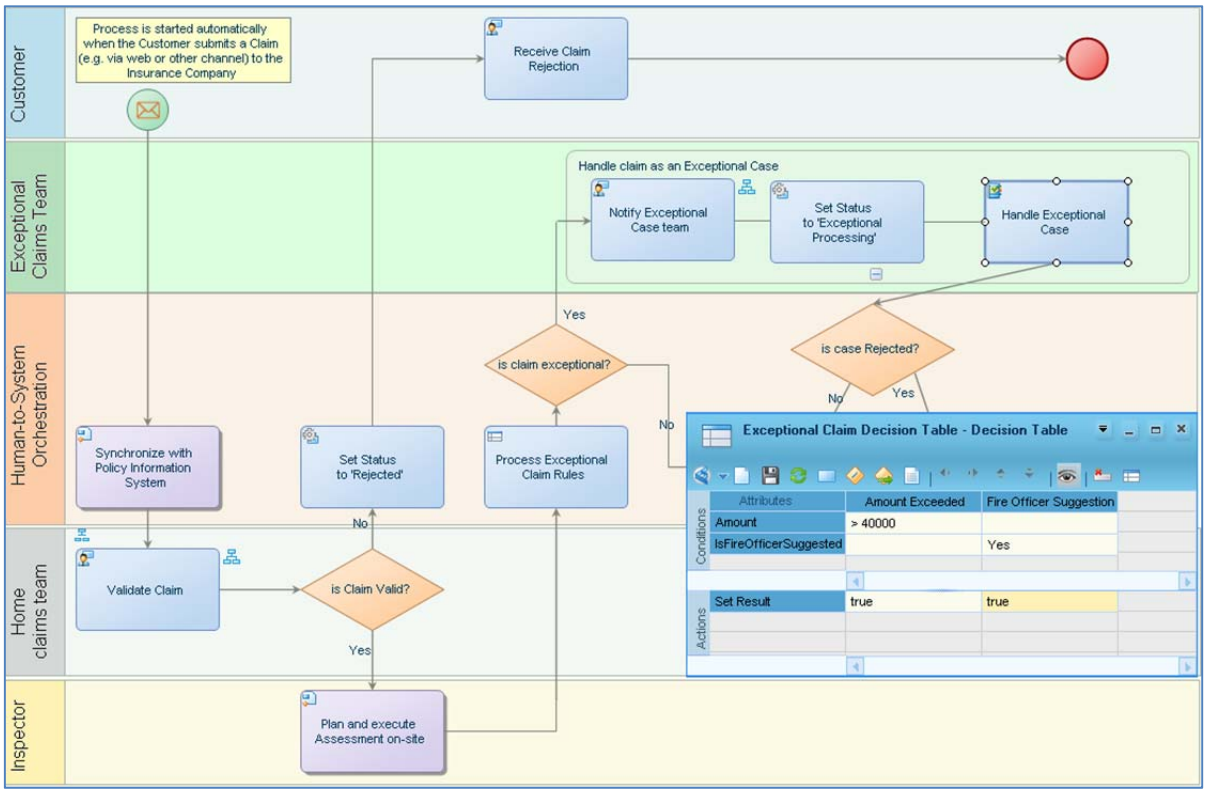
Interstage BOP offers a powerful, high performing rule engine for expressing business logic in the form of rules. Business logic (e.g. calculation of discount, calculation of customer rating etc.) is well suited for expressing in business rules, the reason being such logic is prone to frequent changes. Business rules should be easy to modify and deploy by business users. Part of the logic of business processes is often expressed as business rules to make the business process less complex, more flexible and easier to maintain.



Rule engine architecture

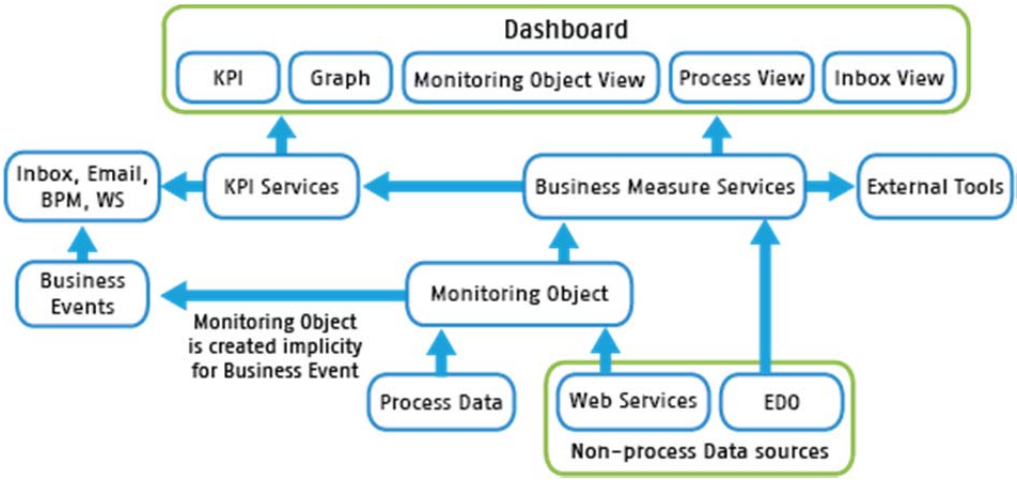
- The application developer decides which logic should be part of Java code and which logic should be expressed as rules. The rule of thumb is that any logic which is very dynamic and changes frequently should be expressed as rules
- Decision tables in Interstage BOP are layered over the concept of rules. They abstract a user from the procedure of building a rule from scratch. A decision table can be created based on a business object; each decision table can contain one or more rules acting on that business object. Decision tables provide a concise, easy-to-read, highly maintainable and logically organized way of representing and querying data.
- Decision tables can be used directly as an activity in a BPM.
- Alternatively, you can generate a web service on top of a decision table, thus enabling access to it from all web services consumers.
- The Interstage BOP rule engine is available as a Java library, allowing any application to use it as an in-process Java library.

The following screenshot shows the decision table being invoked from the Main BPMN flow for the insurance claims handling example.



The Decision table is invoked as a web service from the BPMN flow, the blue BPMN activity labeled 'Process Exceptional Claim Rules' is in fact the invocation of the decision table, executing at runtime its Conditions and Actions (if-then-action rule)

BUSINESS ACTIVITY MONITORING



BAM architecture overview

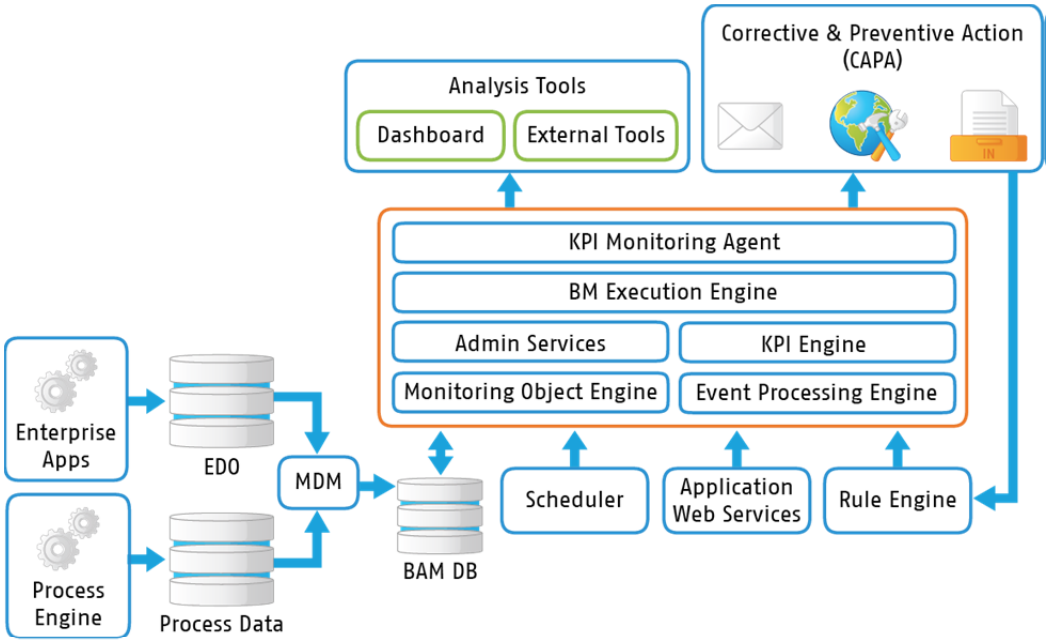
Business Activity Monitoring (BAM) aids in monitoring key business events for changes or trends indicating opportunities or problems on which business managers can take corrective action.

Interstage BOP BAM provides real-time alerts and notifications of critical events and a centralized performance dashboard. It offers a drill down analysis to discover trends, patterns and bottlenecks.

In addition to monitoring business processes, BAM can also be used to monitor data coming from other systems using Interstage BOP **Master Data Management** (MDM) or web services and present it to users via alerts or on the dashboard.

To elaborate further on how it works:

- The data for both EDO and processes is collected through MDM. The process data is picked up by identifying process events based on process state transitions in the database.
- The business measure web services are executed by fetching and executing corresponding SQL query from the BAM metadatabase.
- KPIs are monitored by the monitoring agent (scheduler) by invoking KPI web service with parameters specified in the KPI editor during design time.
- The BAM dashboard charts are built through the XForms designer, using FusionCharts (Adobe Flash-based) graphical controls.



BAM runtime architecture

WS-APPSERVER

Next to processes and user interfaces, applications often leverage an application server to deliver business entities and accompanying logic. This is done through Interstage BOP WS-AppServer; it enables the application to define its core application and UI logic in a simple and structured manner. Interstage BOP WS-AppServer pulls out database metadata from a relational database and generates Java Code.

The generated code along with the WS-AppServer framework provides the application with the following features:

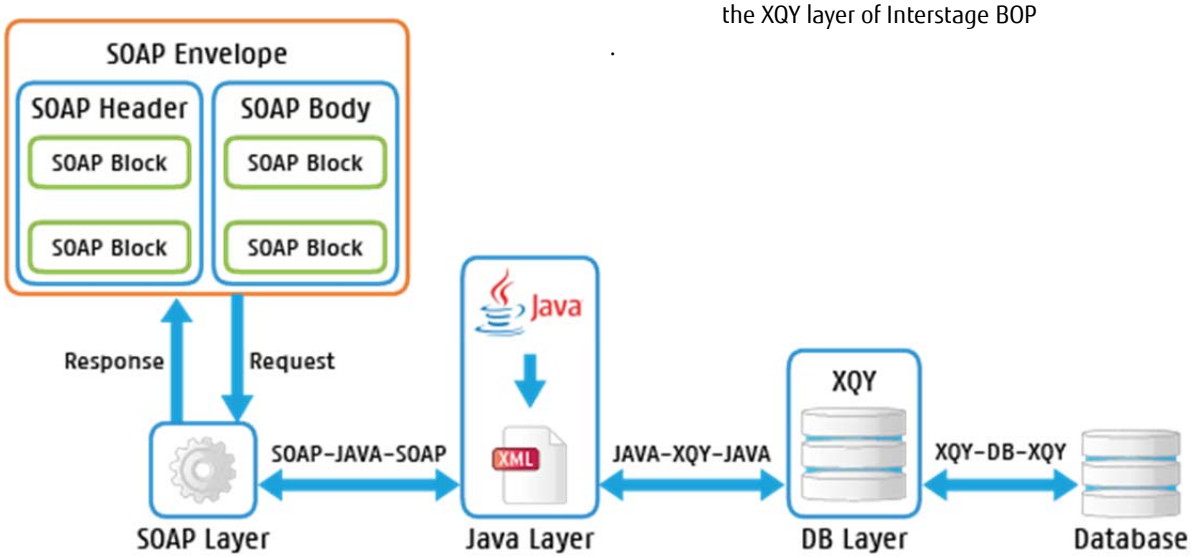
- Rapid Application Development, because of a reduced need for coding due to the model-driven approach.
- Persistence is completely taken care of by the framework and there is no need to write code to perform CRUD operations.
- Transaction management is handled by the framework.
- Object Life Cycle management is handled by the framework.
- Event-based programming model makes coding easier.
- WS-AppServer has a large number of object life cycle and transaction related events which are raised by framework; based on its specific needs, the application just needs to fill in what it needs to do when a certain event is raised.
- Support for changing dynamic business logic easily via integration with the rule engine.
- Logic that is unlikely to change frequently is put inside the Java code.
- Logic that changes frequently is defined as business rules, thus ensuring that business users can change the application behavior without having to bring in their developers to change the code.
- Out of the box support for exposing the application logic as web services.
- Support for aggregating database specific classes into user defined classes, called custom classes, that map closely to the 'view' the end user needs to see.
- Support for extending business logic of non-Interstage BOP systems by defining classes over entities of external systems and using the WS-AppServer events to handle the persistence and define the business logic.

Since WS-AppServer is a part of Interstage BOP, many features that an application needs like authentication, authorization and scalability come for free, though they are not provided by the WS-AppServer component.
These features ensure that the application has to focus only on its core logic and the rest is left to WS-AppServer and Interstage BOP.

The WS-AppServer framework has three distinct layers:

- **SOAP messaging layer**
  - The SOAP messaging layer manages the transaction
  - It processes the incoming SOAP request, creates the corresponding Java Objects, calls the setters to fill the object with the information coming from the SOAP request and then calls the desired operation on the object as specified in the SOAP request
  - Once the request processing is done, it creates the SOAP response from the Java Object.
- **Object layer**
  - This layer manages the object data as XML. As XML is the common language in Interstage BOP, it is easy to communicate to other Interstage BOP components.
  - This layer manages the object life cycle and calls the object life cycle events.
- **Persistence layer**
  - This layer persists the data in relational database, using the XQY layer of Interstage BOP

.



WS-AppServer architecture

SCHEDULING

In a real-time business environment, the ability to trigger processes or applications based on either specific system events or at a specific time is a critical requirement. Interstage BOP facilitates modeling of schedules that can trigger time-based actions like processes or web service invocations.

Interstage BOP provides an intuitive UI-based schedule modeler, enabling modeling of different kinds of schedules that can be integrated into an application.

Schedules are broadly of two kinds:
One-time schedule: These are executed only once
Repeating schedule: These are triggered at specified periodic intervals, ranging from annual to hourly

At the scheduled interval, one of the following actions can be triggered:

- Invoke a web service call, by providing appropriate parameters
- Instantiate a new business process by providing the process model name and the input message
- Call back an already running business process instance by providing the instance ID.

One-time schedule:

Type of Schedule	Description
Runnow	Once created, this schedule is instantly executed
Duration	This schedule can be set to execute after a specified duration

Repeating schedule:

Type of Schedule	Description
Hourly	These schedules can be set to recur at a specified time every hour
Daily	These schedules can be set to recur at a specified time every day
Weekly	These schedules can be set to recur at a specified time every week
Monthly	These schedules can be set to recur at a specified day every month
First Day of Month	These schedules can be set to recur at a specified time on the first day of every month
Last Day of Month	These schedules can be set to recur at a specified time on the last day of every month
First Week Day of Month	These schedules can be set to recur at a specified time on the first week day of every month
Last Week Day of Month	These schedules can be set to recur at a specified time on the last week day of every month

TASK AND NOTIFICATION SERVICE

Every Interstage BOP user is assigned an Inbox, which is used to send and receive tasks and notifications. The inbox is configured for a given user profile and functions like a mailbox. Interstage BOP users can access tasks that are sent to their work lists, to the roles they are associated with, to the teams they are part of or to their personal task list.

TASKS

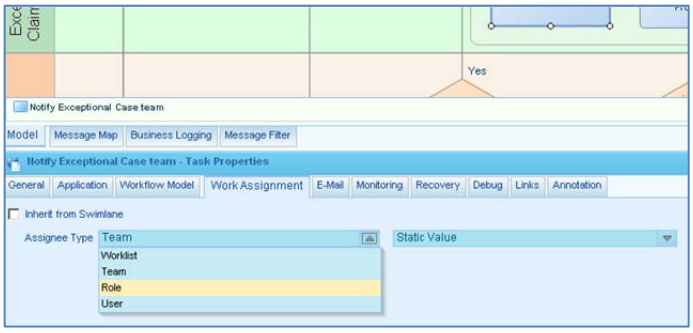
A task is an activity of a process that is to be executed by a human participant of the process. Users can either respond to the task, add a new action to the flow after the task, or do both.

For example, when the stock of a particular item in a warehouse reaches a certain level, a replenish task is sent to the warehouse manager. On receiving the task, the warehouse manager opens it, fills in a purchase order form and forwards it to the purchase manager.

NOTIFICATIONS

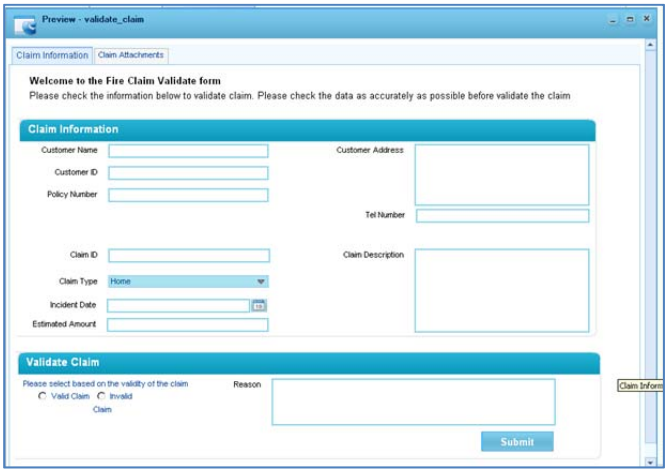
Notifications are used to convey event information to designated recipients or roles. For example, a notification can be used to send a message to the Exceptional Claims Team that there is a claim to be handled as an exceptional case.

The screenshot in the following diagram shows how the work assignment can be configured.



Configuring work assignments via the properties tab

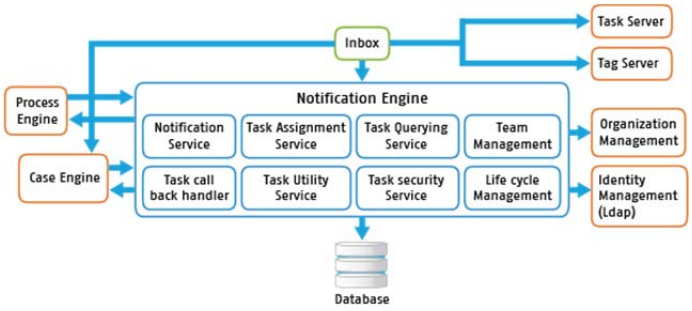
The following diagram is a screenshot of Interstage BOP User Interface used for one of the tasks of the earlier example of an insurance claim BPMN process model.



Previewing a user interface from a business process activity

The user interface is shown in EDIT or PREVIEW mode when clicking the user interface icon part of the Process Activity Validate Claim in the process model. This is another example of Interstage BOP' model-driven approach: models for flows, for cases, for task user interfaces, etc.

The life cycle of both tasks and notifications are managed by a service container called Notification service.



Notification service architecture

MASTER DATA MANAGEMENT

Interstage BOP **Master Data Management** (MDM) focuses on the core infrastructural needs of enterprise data integration. Interstage BOP separates infrastructural MDM needs from all that is domain-specific.

This implies a data domain-agnostic approach to data integration. Users can apply Interstage BOP MDM for integrating data of different data types - master, reference and transactional data. In addition, within a given data type (e.g. master data), Interstage BOP MDM can be applied to harmonize data of different subject areas such as Customers, Suppliers, Products, Locations etc.

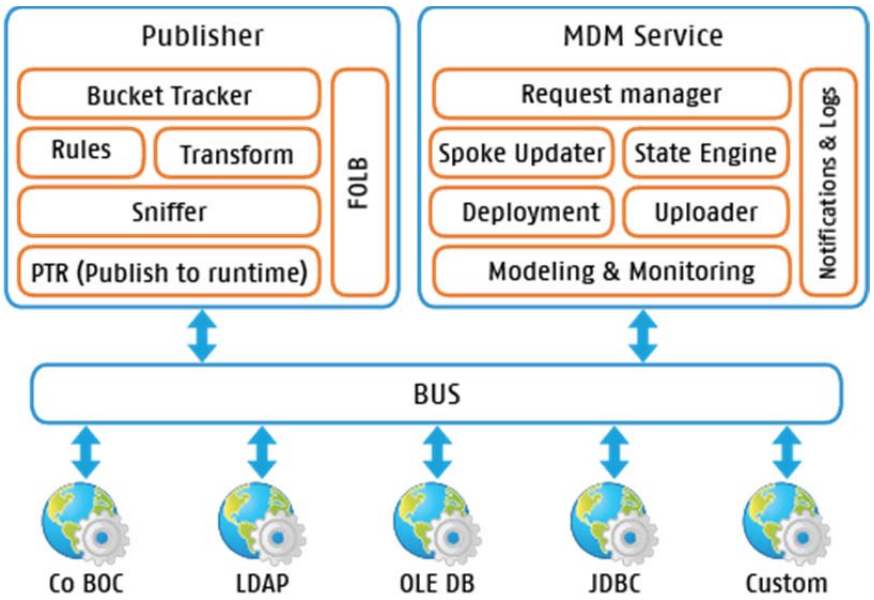
Highlights of Interstage BOP MDM offering:

- Pluggable architecture (employing third party data quality tools etc.)
- Near real-time data synchronization
- Master data, business object life cycle management
- Event driven object life cycle management
- Strong workflow capabilities
- Works in publish - subscribe model
- Support for web services
- Supports all 3 MDM patterns (Registry, Co-existence, Transactional)

Thus, there are a wide variety of ways in which organizations can utilize Interstage BOP MDM to build and deploy trusted data hubs, according to the needs and goals of the enterprise.

In summary, this is what MDM can provide for the enterprise:

- Identify sources of master data
- Identify producers and consumers of master data
- Collect and analyze metadata of master data
- Appoint data stewards
- Implement a data governance program
- Develop a master data model
- Design and setup the infrastructure
- Generate and test the master data
- Modifying producing and consuming systems to integrate with the new MDM solution.
- Implement a maintenance process



MDM runtime architecture

SERVICE ORIENTED ARCHITECTURE LAYER

INTERSTAGE BOP XML TECHNOLOGY

From its inception, Interstage BOP uses XML as its core communication language. The information exchange across Interstage BOP components is in the form of XML, which leads to heavy XML processing. This necessitated the development of better implementations of XML processing technologies like XML parser, XPath/XSLT processor.

Interstage BOP XML parsing API has 2 flavors. The first flavor, **Native Object Model** (NOM), is a set of proprietary APIs, which is highly efficient but can be a bit complex in use. The second variant, coined DOM over NOM, is a standard DOM API layered over the NOM APIs, bringing the benefits of a convenient standards compliant API while keeping the overhead extremely low. Developers can choose their favorite API flavor and even switch from one to the other.

ENTERPRISE SERVICE BUS

The Interstage BOP strategy has always been to develop an integrated suite of services to enable customers to develop, deploy and manage business applications on a Services Oriented Architecture (SOA). Therefore, Interstage BOP has a very broad view of middleware that forms the platform on which the other components are built and offered as services. These are then consumed and orchestrated by our customers in a process driven approach for building their custom applications, their web applications and their composite applications.

The center of this middleware is the enterprise ready Interstage BOP Enterprise Service Bus (ESB), which is the enabler for SOA. This unique technology enables every service as a loosely coupled, distributed service on a bus, with the associated benefits of granular fail over and scalability.

Traditional ESB implementations have emerged from Message Oriented Middleware (MOM) by adding web services and Enterprise Application Integration (EAI) on top of this existing MOM infrastructure. In a hub-and-spoke architecture, all back-end systems (spokes) rely on the hub to communicate with each other and any hub failure causes the entire integrated system to fail. In addition, any back-end systems can potentially overload the hub, making it necessary to augment the hub with additional computing power.

Interstage BOP ESB does not have a central hub; this eliminates the single point of failure and removes a common performance bottleneck. Interstage BOP ESB uses a bus as its architecture and 'peer-to-peer' as the communication paradigm.

The enterprise service bus has emerged as the best practice to perform application connectivity and to decouple service consumers and service providers, and is today serving as the most popular software infrastructure for SOA.

The following focuses on some of the features and services which are expected to be delivered by any modern day ESB, and describes how Interstage BOP ESB provides them.

STANDARDS COMPLIANCE

Interstage BOP uses universally accepted standards such as WSDL, XSD, XML and SOAP and all consumers of Interstage BOP web services can use the "design by contract" model for invoking these web services without having to worry about underlying implementations. Interstage BOP also is WS-I Basic Profile compliant, so the services hosted on Interstage BOP are completely compatible for consumption from different platforms. Developers can base their functionality on the contract given by the WSDL and not worry about implementations.

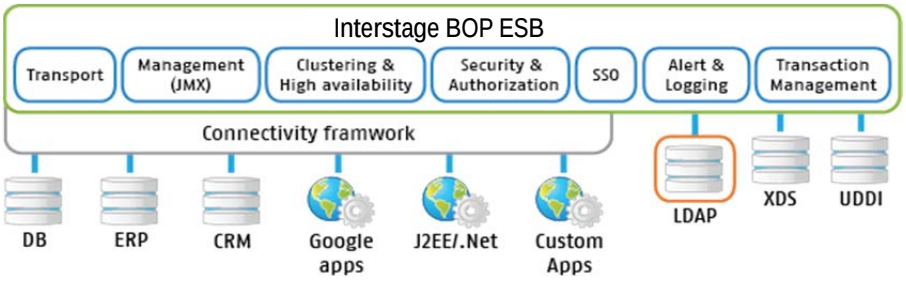
LOOSE COUPLING

All capabilities provided by Interstage BOP or built by customers using the platform are delivered to the end user as services and from a technology perspective as web services. Web services use WSDL and XSD, and provide loose coupling because they formalize the contract between the consumer and provider. Interstage BOP messages follow the paradigm of stateful objects and stateless connections, so all invocations for Interstage BOP services are completely decoupled and do not hold any information about the clients' prior interactions.

TRANSPORT TRANSPARENCY AND MULTIPROTOCOL SUPPORT

A critical ability of the ESB is to route service requests through a variety of communication protocols and to transform service requests from one protocol to another where necessary. Enterprise applications typically involve talking to different individual applications which support different transport protocols, and Interstage BOP ESB provides physical transport protocol bridging to allow communication between services using different transports.

Interstage BOP provides a choice of configuring services on sockets, Microsoft based Message Queues (MSMQ), and Java based Message Queues (JMS) and also offers the flexibility to build custom transports. This gives flexibility to effectively integrate disparate systems and manage complex communications at the transport level.



Enterprise service bus facilities

LOCATION TRANSPARENCY

The ESB acts as a layer of middleware through which a set of business services are made widely available, but the client does not or should not be aware of the physical location at which the service is hosted as it breaks the loose coupling strategy. All Interstage BOP services point to one physical address and Interstage BOP ESB locates the service when it is invoked, providing a level of service virtualization and location transparency so that if a machine goes down or a service provider has to be moved, individual service clients do not need to be notified of the change.

CLUSTERING - LOAD BALANCING AND HIGH AVAILABILITY

The ESB acts as the middleware to host all services. It is also well suited to perform load balancing of service requests across services. Interstage BOP services are configured under a logical entity called service group, each service is hosted in a service container, which is the Java process providing the implementation. A service group can be implemented through more than one service container of the same configuration, each running on a different physical machine. An administrator can choose the load balancing algorithm such as simple fail over or Round Robin for load balancing. Service containers can be added or removed on demand depending on the need. Interstage BOP clustering leverages a reliable broadcasting technology called Gossip. Service containers broadcast their state and health information using the Gossip protocol.

SECURITY INFRASTRUCTURE

As the ESB acts as the central mediator for service invocations, it is ideal for declaring and enforcing security. Interstage BOP ESB provides a comprehensive security infrastructure for developers to create access control lists for specific roles/users on all important components such as service groups, web services and metadata, which are then enforced by the engine after authentication. Interstage BOP can integrate with any enterprise domain authentication systems, integrate with external SAML providers and also provide the option of custom authentication.

RELIABLE MESSAGING

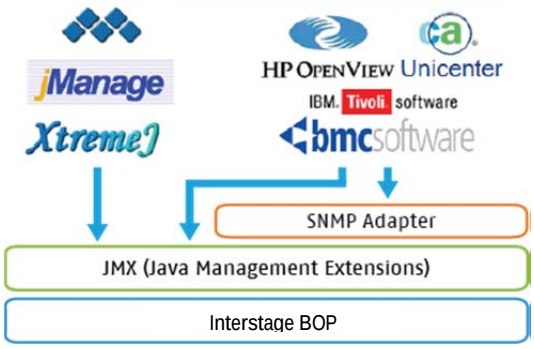
Reliable messaging refers to the ability to queue service request messages and ensure guaranteed delivery of these messages to their destinations. It includes the ability to provide message delivery notification to the message sender/service requester. Interstage BOP ESB supports distributed transactions and Interstage BOP components are configured for accepting requests on message queues. They can deliver and receive messages reliably, even in case of component, system or network failures. Interstage BOP's reliable messaging uses distributed transactions through the XA protocol and supports JMS and MSMQ message queues

MANAGEABILITY

It is important to monitor the health of the ESB and the services hosted on it. Interstage BOP ESB comes bundled with applications which provide comprehensive information about all the services, health of the system in a dashboard view, and the ability to drill down to specifics for any component. Interstage BOP has self-healing capabilities, where services raise alerts and can take corrective actions automatically.

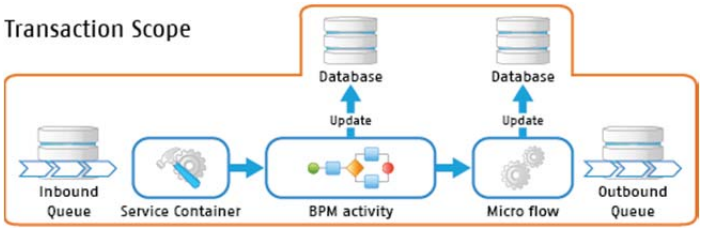
Interstage BOP services can be monitored using any JMX and SNMP compatible tools.

- **Manageability using JMX**
  Interstage BOP can be managed through the JMX APIs as exposed by the platform. JMX-capable tools can directly interact through JMX. Other tools can use an SNMP adapter.



Management interfaces

- **Composite Application Logging**
  Interstage BOP ESB supports diagnostic logging, known as Composite Application Logging (CAL) for composite applications. The composite application log messages have contextual information like correlation ID and multilevel diagnostic contexts, which indicate the execution path of the application. The context is non-intrusively propagated across several layers based on the execution. This enables administrators to perform causality analysis of any incident. The log messages are optionally stored in a centralized database, thus providing an integrated view of the logs of a cluster of multiple nodes. The composite application log viewer provides drill down, correlative analysis and filtering.



Transaction spanning reliable messaging, BPM and application updates

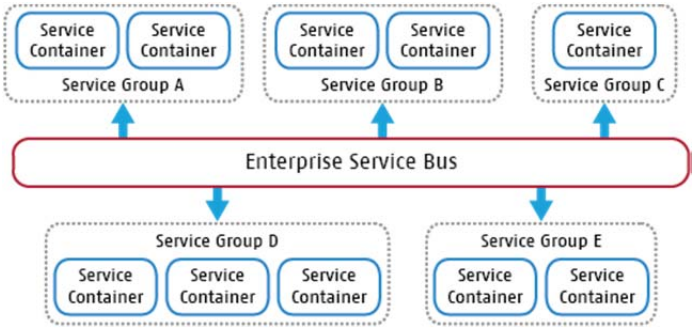
IMPLEMENTATION ASPECTS OF INTERSTAGE BOP ESB

RUN-TIME NOTION

Web services are served and processed by service containers. A service container is a Java virtual machine, typically running as a physical process on the system.

Service containers are logically grouped as service groups. Service groups hold the necessary configuration information to identify the type of request and the routing. Requests are received through connection points configured on the service containers. Service containers can be configured on multiple machines and each service container has multiple connection points, potentially with different transport protocols. Connection points are end points for services on the ESB.

Service groups provide the loose coupling aspects in terms of location transparency. Connection points enable network protocol neutral message delivery. Service groups, service containers and connection points are configured in Interstage BOP LDAP repository.



Service groups and service containers

MESSAGE ROUTING AND DELIVERY PARADIGMS

All the requests (web service invocations) are identified through the namespace. The namespace indicates the type of request that needs to be processed.

- Based on the request, the respective service group is first identified using Interstage BOP LDAP repository.
- Then based on the availability and routing policy, a specific service container is chosen for processing the request.
- After selecting the service container, the connection point information is used for choosing the transport protocol.
- Request is delivered to the service container for processing.

Interstage BOP ESB supports both synchronous (request-reply) and asynchronous (fire-and-forget, request-callback) messaging paradigms.

INTERSTAGE BOP MONITOR

Interstage BOP Monitor is a special service container on the ESB. It acts a supervisor for all service containers configured on that particular node.

Each node in the ESB cluster has its own Interstage BOP monitor configured and running. It runs as an OS level system process and takes care of the following

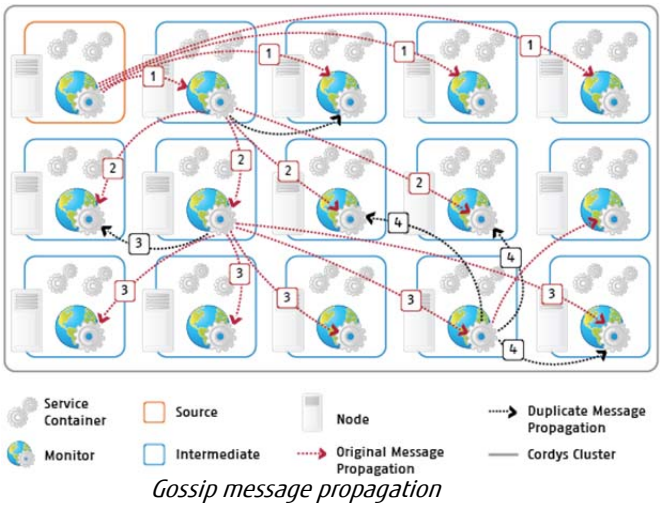
- Managing (start, stop, restart, reset) service containers.
- Providing the necessary bootstrap configuration.
- Synchronization and broadcasting of status information of all service containers on the cluster.
- Synchronization and broadcasting of any problems faced by any service on that node.

GOSSIP - INTERSTAGE BOP CLUSTERING TECHNOLOGY

Interstage BOP is a distributed system that uses multiple processes (Java virtual machines) spread across machines. Interstage BOP ESB uses routing and load-balancing techniques to distribute user requests to appropriate processors (called service containers). These activities require group communication to be included in the ESB, for which the Gossip protocol is chosen.

Some of the use cases for this framework are

- **Distributed cache invalidation**: Most of the components (e.g. LDAP, Rule, and CoBOC) in Interstage BOP use caching for performance reasons. All these components use Gossip for cache invalidation.
- **State registry**: State registry is an in-memory database of the state of Interstage BOP cluster, built on Gossip. Interstage BOP state registry provides the needed infrastructure for high-availability.



Gossip message propagation

The Interstage BOP ESB connects a set of loosely coupled web services to form a Smart Services Grid. This grid is web services based: all messages are in the SOAP format and are described in standards compliant WSDLs. The deployment options vary from a simple test setup on a medium laptop till a large distributed cluster that scales to a very high number of requests per second. Such a distributed system can be setup with no single point of failure, thus providing a fault tolerant system. The Interstage BOP ESB is open and portable: it is supported on Windows, various Linux flavors, AIX and Solaris and can work with various databases: Oracle, Microsoft SQL Server and MySQL

APPLICATION PACKAGING AND DEPLOYMENT

Interstage BOP provides both web based and command line based facilities for managing the applications built using Interstage BOP. Any composite application developed using Interstage BOP is packaged through a format called Interstage BOP Application Packaging (CAP). Packaging the application is part of standard CWS facilities.

A major part of the platform is packaged as CAPs and deployed through a CAP deployer. The CAP format is well defined to manage dependencies, prerequisites and cluster level deployments for installation, upgrade and uninstallation. CAP contains a manifest of all artifacts that are bundled. The manifest maintains the type of information, identification and a hash for each artifact.

- The type of information is used to identify the needed deployment logic.
- It ensures the uniqueness of artifact identification.
- Hash helps to identify the change in the artifact across different versions of the application.

Using the manifest, CAP deployment provides the fine grained detail and impact analysis caused by the application deployment.

For example: The impact analysis section displays the list of artifacts that are affected and whether they are deployed at cluster level or node specific.

Interstage BOP maintains an XDS-based repository called CAP Registry to register the details of all the applications deployed. CAP provides a single step to install, upgrade, uninstall and rollback any application across the cluster.

Optionally, CAPs can be digitally signed to build the chain of trust. The trusted entities are registered through the Security Admin Console. It is configurable to allow / disallow unsigned and tampered applications. This option establishes the trust factor for multitenant deployment over the cloud.

ENTERPRISE INFORMATION SYSTEM CONNECTIVITY

Every enterprise uses one or more business applications and IT systems to manage the business of the enterprise. Solutions developed with Interstage BOP nearly always integrate with existing Enterprise Information Systems (EIS). These applications and systems include ERP systems, Content Management Systems, CRM systems, databases, etc. Interstage BOP provides a generic connectivity framework to connect to various systems and applications. Based on this framework, Interstage BOP and the community around Interstage BOP have developed a set of ready-made connectors for some of the most commonly used IT systems.

Interstage BOP connectors act as an interface between the ESB layer and a specific application, system or technology. It provides a two way communication channel. Requests or messages from the ESB layer are converted to a format which is understandable by the specific application or system and messages from the application are converted to SOAP requests on the ESB.

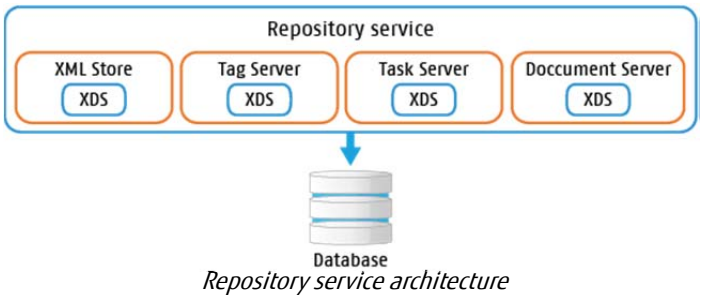
The application connector framework leverages the rich functionalities and features provided by Interstage BOP, which include:

- Logging
- JMX support
- Access control list (ACL)
- Problem Registry
- Localization support
- Transaction support
- Load balancing and fail over support

The generic connectivity framework helps ISVs and application developers to develop specific connectors with minimum effort, and also enables the third party connectors and adapters to be integrated with Interstage BOP.

REPOSITORY

Interstage BOP consolidates storage of metadata in a single repository called XDS. This repository is the underlying store for Interstage BOP Collaborative Workspace (CWS) but also stores tag definitions, tasks, business calendars, etc. XDS is RDBMS-based and meets the high availability requirements of Interstage BOP.



Repository service architecture

Interstage BOP repository is populated from the following sources:

- **Application content developed in CWS** - Stored in XDS via CWS.
- **On publish of application content from CWS** - Deployed models are stored in XDS (example Business calendar, organization models, Tasks etc.)
- **Runtime data** - Applications/platform can store metadata and definitions in the repository (e.g. XMLStore).
- **Platform metadata** - stored in repositories. Package installation details, for example, are stored in XDS.

The ready-made connectors provided by Interstage BOP and the community around it provides instant access to most widely used systems in an enterprise. Some of the out-of-the-box connectivity options provided by Interstage BOP are:

- **Database connector**: This is used to interact with database (RDBMS) systems using popular database technologies like JDBC and OLEDB. This layer acts as an XML to relational mapping layer and vice versa.
- **E-mail connector**: The e-mail connector enables sending and receiving mails through IMAP or SMTP and POP3.
- **SAP connector**: SAP connector provides connectivity to SAP back-ends through RFC and BAPI.
- **Script connector**: Script connector allows web services to be implemented in JavaScript.

TAGGING

Many social networking sites nowadays allow their users to tag the information. Tags are generally chosen informally and personally by the item's creator or by its viewer, depending on the system. Tagging is an easy way for the end user to group/categorize their own tasks and artifacts. Tagging abstracts the storage approach (e.g. hierarchical) and enables users to view the system in a more linear and self-managed approach.

The tagging service within Interstage BOP allows associating tags to any type of artifact. Interstage BOP currently uses it to associate tags to tasks on Interstage BOP User Start Page (CUSP), tasks in the inbox, and documents in the Interstage BOP Collaborative Workspace (CWS). The design is extensible and allows tagging application artifacts (e.g. orders, products, employees) as well.

AUDITING

Auditing is a well-known process of tracking changes to an object of concern in software. Interstage BOP provides an auditing framework which is used internally by many of its components and can be used for auditing application events as well.

Interstage BOP provides an auditing framework to define artifact types and provides needed abstraction to help developers audit the needed artifacts. Interstage BOP stores all the audit information in the Interstage BOP database configured during installation.

The audit framework is used to audit several deployment activities, user management, changes to rules, invalid SAML assertions, etc.

GATEWAY

Interstage BOP web gateway is the HTTP interface of Interstage BOP Smart Services Grid. It is meant to be a light-weight component; hence its functionality is very minimal. In brief, it is the HTTP end-point of all web services hosted by Interstage BOP. Besides that, it also supports some security features.

The primary functions of the web gateway are:

- Authentication (optional)
  - Integrated authentication, e.g. Active Directory
  - Interstage BOP authentication, using Single Sign-On (SSO) service
- Authorization. The set of accessible web services can be limited on web gateway level.
- SOAP request/response validation (optional)
  - Verify if request is according to the WSDL
  - Verify if response is according to the WSDL
- Translate HTTP requests into SOAP requests
- Translate SOAP responses into HTTP responses

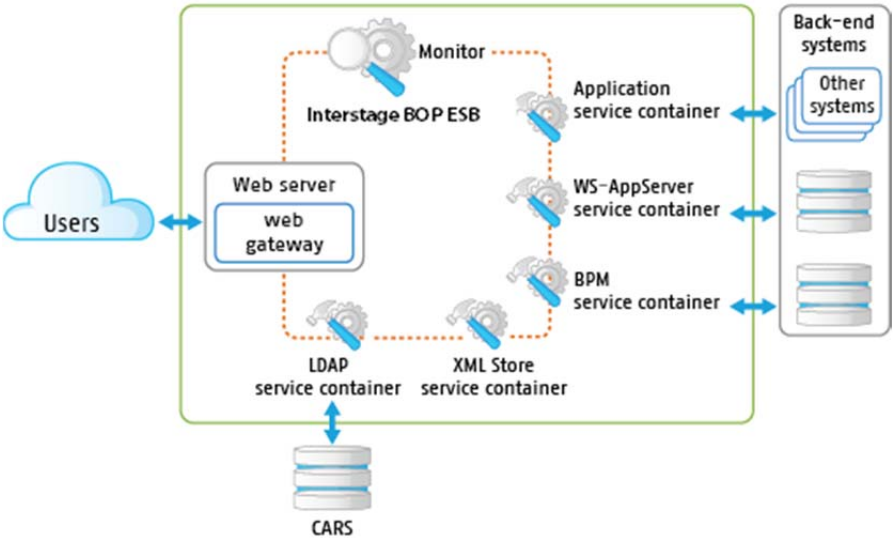
USER INTERFACE (UI) TASKS

UI applications in Interstage BOP are referred as UI tasks. UI tasks facilitate fine grained authorizations on different elements of the UI.

The UI task modeling environment allows identifying different elements as task parts. The task parts may be linked to any web service invocations, or just UI elements, or a combination of both. The UI tasks are assigned directly to users or through roles. During this assignment, fine grained authorizations can be granted by enabling or disabling different task parts. The assigned UI task is called 'configured task'. It takes care of updating all access controls, for a particular user or role, which are required to invoke the UI task.

For example: An administrator can allow certain users to start and stop service containers, and limit other users to only view the status of service container. Both groups will have the same UI with some options enabled or disabled. Enabling and disabling is done through configuration, not during the development time. Bypassing the user interface does not breach security, as the related backend authorizations are granted along with the UI permissions.

It is possible to define composite UI tasks. UI tasks can be linked to workflow tasks through the inbox.



Positioning of the web gateway

SECURITY

AUTHENTICATION

Authentication is about establishing the identity of the user within the Interstage BOP system in a secure and trusted way.

Interstage BOP has multiple options for user authentication:

- Web server authentication
- Interstage BOP authentication
- External authentication

WEB SERVER AUTHENTICATION

With Web server authentication, the responsibility of authenticating the user is put at the web server. The web server will handle user authentication and pass-on the user identity to Interstage BOP.

Web server authentication can be handled in different forms:

- Basic, Digest, Domain Authentication (NTLM) and
- Certificate-based authentication.

Certificate-based authentication adds an extra level of security to the platform as PKI is used to identify the user. Each user has a unique certificate which will be used for authentication in the web server. Only after the client certificate is valid, the user identity is passed on to Interstage BOP.

INTERSTAGE BOP AUTHENTICATION

With Interstage BOP authentication, the user authentication is done by Interstage BOP. The web server will not do any authentication of the user; this is left to Interstage BOP Single Sign-On service (SSO).

User authentication is done based on a username and password and after validating them, SSO generates a signed SAML 1.1 This SAML 1.1 Assertion includes a SAL artifact which can be used as a reference to the SAML Assertion.

With each web service request from the front-end, this SAML artifact communicates in a web-browser session cookie. The web gateway will look up the corresponding SAML Assertion for the given SAML artifact and use this internally in Interstage BOP.

When a user logs into Interstage BOP, the username and password are entered in a login form and sent to Interstage BOP SSO service. The username and password, by default, are validated against Interstage BOP Administration Repository Server (CARS).

Interstage BOP Authentication is extensible in a way that custom login forms can be used in the front-end. Interstage BOP SSO service also has an extension mechanism, so that one can also authenticate against other sources (e.g. Active Directory or a database).

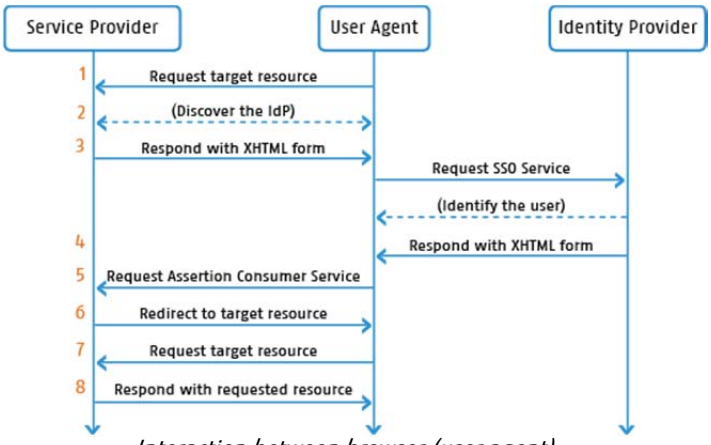
As complete user authentication is done by Interstage BOP, authentication in the web server needs to be disabled.

EXTERNAL AUTHENTICATION

Another form of non-web server authentication is what is called external authentication. Here, authentication is relayed to an external service or Identity Provider (IDP). Interstage BOP implements the SAML 2 Web browser SSO profile, which means the external IDP must support the SAML 2.0 standard.

When Interstage BOP needs to authenticate the user, the browser will be redirected in a separate window to the configured external IDP which handles the authentication. After the user is authenticated, the external IDP Posts a 'SAMLResponse' back to Interstage BOP SSO service. The SAMLResponse contains a signed SAML 2.0 Assertion which states the identity of the user. This external SAMLResponse will be validated and matched with the available users in Interstage BOP. After validating the external SAML 2.0 Assertion, a new internal SAML 1.1 Assertion is generated. So the external SAML 2.0 Assertion is only used once for user authentication.

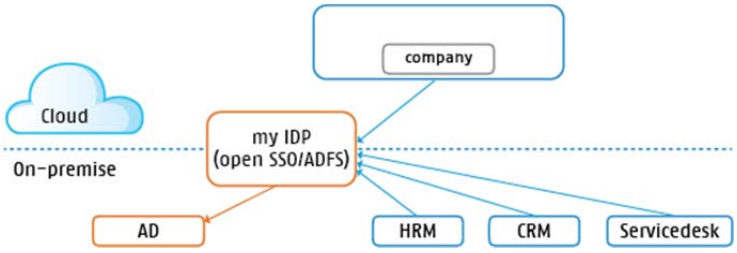
The flow between the browser, Service Provider and Identity Provider is given below. In this flow, Interstage BOP SSO is the Service provider and the User Agent is the user's browser.



Interaction between browser (user agent), SSO (service provider) and identity provider

The external authentication is based on a two-way trust configuration between Interstage BOP and the external IDP. The Security Admin Console in Interstage BOP is used to configure one part of this trust. See the Security Management space for documentation and examples.

The external IDP might be cloud-based or on-premise. Direct communication between the external IDP and Interstage BOP is not required. The information exchange is done through browser redirects and automatic form submits only.



The external IDP can be cloud-based or on-premise

ADVANCED AUTHENTICATION FEATURES
Advanced authentication features like password policies, strong authentication, One-Time-Passwords (OTP) are supported through the external IDP.

When a Single Sign-On (SSO) user experience is needed, then this can be implemented with configuring multiple applications with the same Identity Provider. When all involved applications share the same Identity Provider, then the user only needs to authenticate once at the Identity Provider and not on a per application basis.

AUTHORIZATION
Interstage BOP has a role-based access system. Roles are assigned to users, and a set of privileges are defined, formulated in Access Control Lists (ACLs). Roles can be composed of other roles.

ROLES
- **ISV Role**: Role defined during application development, defining access to web services, data and tasks. This is packaged as part of an application package and loaded to ISV level. A call handling application could, for instance, introduce an Escalation Manager role.
  The following role types exist:
  - *Is functional*: indicates whether the role is to be shown in organization model.
  - *Is internal*: indicates it should be skipped in the organizational modeler and user manager.
  - *Is technical*: indicates it should be skipped in the organizational modeler but shown in the user manager.
- **Organizational role**: Role defined during run-time, normally aggregating ISV roles. This is created on an organization level. An organization could define the roles *Project Manager* and *Development Manager* and assign the application role *Escalation Manager* to both.

ACCESS CONTROL LIST
The ACL contains a set of authorizations or permissions that are added to a role or user. Each time a user accesses a protected resource, (e.g. a web service), an access control request is formulated and matched against the complete set of ACLs for this user.

WEB GATEWAY SECURITY FEATURES
The sandboxing feature of the web gateway can be leveraged to enhance the security of an Interstage BOP system. It is implemented as part of the ISAPI extension and the Apache module running in the web server.
Sandboxing is a mechanism which restricts the SOAP requests and tells them when they can be executed on a web server. It is configured through Interstage BOP Management Console on the server.

CONCLUSION

In this white paper, we have seen that Interstage Business Operations Platform offers a strong foundation to build business applications. The white paper described the design goals of the platform and provided insights into the design-time architecture as well as the runtime architecture of the platform.