TECHNICAL SPECIFICATION

SOA Blueprints Reference Example Requirements Specification

Draft v0.5 (For Public Review)

A set of interconnected applications demonstrating the Blueprints and Best Practices for SOA

http://www.MiddlewareRESEARCH.com

The Middleware Company
Research Team

Steve Wilkes & John Harby
June 2004

research@middleware-company.com

Expert Review Group Members:

Will Pugh (BEA), Paul Patrick (BEA), Steve Jones (Cap Gemini),
Hugh Grant (Credit Suisse First Boston), Chris Peltz (Hewlett-Packard),
Scott Metzger (Truelink), Bola Rotibi (Ovum), Annrai O’Toole (CapeClear),
Frank Cohen (PushToTest), Alan Zenreich (Ness Technologies),
Michele Leroux Bustamante (IDesign), Vince Salvato (E2E Consulting), David Murrell (Xede),
Rob Castaneda (CustomWare), Doron Sherman (Individual), Chris Judson (E2E Consulting),
John Cheesman (7irene), Rainer Ammon (FH Wels), Mike Sawicki (Compuware),
Shigeru Urushibara (UL Systems), Akira Hirasawa (UL Systems Inc), Koichi Hayashi (UL Systems)
# TABLE OF CONTENTS

1 OVERVIEW ................................................................. 6
2 PROVISION OF AN ENTERPRISE WIDE SECURITY SERVICE .......... 7
   2.1 Overview ............................................................... 7
   2.2 Security Data Structures ........................................... 8
   2.3 HR Publishing to Security System ................................... 9
      2.3.1 The HR Change Topic ........................................ 11
      2.3.2 Employee Change message ................................... 11
      2.3.3 Roles Change message ........................................ 12
   2.4 The Legacy Adaptor ................................................ 12
      2.4.1 Subscribe to the Security System ............................ 13
      2.4.2 Unsubscribe from the Security System .................... 14
      2.4.3 Handle Update Employee ..................................... 15
      2.4.4 Handle Update Employee Roles ............................. 17
      2.4.5 Get Possible Roles ............................................. 19
   2.5 Security System Services ........................................... 20
      2.5.1 User Propagation Services ................................. 20
      2.5.2 User Information Services .................................... 28
      2.5.3 Security Adaptor Services ................................. 35
   2.6 Provision of Security Services through Intercepts ................. 45
3 PROVISION OF A PRODUCT DATA SERVICE .......................... 48
   3.1 Overview ............................................................... 48
   3.2 Data Structures .................................................... 48
   3.3 Services .................................................................. 50
      3.3.1 Get Categories ............................................... 50
      3.3.2 Get Category ................................................... 51
      3.3.3 Get Products ................................................... 52
      3.3.4 Get Product ..................................................... 52
4 CONSTRUCTION OF AN EMPLOYEE PORTAL ..................... 54
   4.1 Overview ............................................................... 54
   4.2 Authenticate .......................................................... 55
      4.2.1 Check Authentication ......................................... 56
      4.2.2 Login / Logout ................................................. 56
      4.2.3 Change Password ............................................. 56
      4.2.4 Role Based Access Control ................................. 57
   4.3 Manage Task List ................................................... 57
      4.3.1 Add Task ......................................................... 58
      4.3.2 Update Task ..................................................... 58
      4.3.3 Get Tasks ......................................................... 59
      4.3.4 Delete Task ...................................................... 59
      4.3.5 Get Task ........................................................ 60
      4.3.6 Portlets .......................................................... 60
   4.4 Browse Organization ................................................ 61
      4.4.1 Find Employee .................................................. 62
      4.4.2 Get Employee .................................................... 63
      4.4.3 Get Departments ............................................... 63
      4.4.4 Get Department ............................................... 64
      4.4.5 Portlets .......................................................... 65
   4.5 Manage Expense Reports ............................................ 66
      4.5.1 Use Case Sequence ........................................... 67
      4.5.2 Data Structures ............................................... 68
      4.5.3 State Diagram .................................................. 69
## 4.5.4 Expense Reporting Services ................................................................. 69
## 4.5.5 Portlet Implementation ........................................................................ 86

### 4.6 Manage Reviews ................................................................................... 92
- 4.6.1 Use Case Sequence ........................................................................... 93
- 4.6.2 Data Structures .................................................................................. 95
- 4.6.3 State Diagram ................................................................................... 95
- 4.6.4 Employee Review Services ................................................................. 96
- 4.6.5 Portlet Implementation ....................................................................... 117

### 5 SUPPORTING APPLICATIONS ................................................................. 121

#### 5.1 HR Application ..................................................................................... 121
- 5.1.1 HR Application Data Structures ......................................................... 121
- 5.1.2 HR Page Flow .................................................................................... 122
- 5.1.3 Maintain Departments ....................................................................... 123
- 5.1.4 Add Department ................................................................................ 123
- 5.1.5 Edit Department .............................................................................. 124
- 5.1.6 View Department ............................................................................. 124
- 5.1.7 Maintain Employees ......................................................................... 125
- 5.1.8 Add Employee ................................................................................. 126
- 5.1.9 Edit Employee .................................................................................. 126
- 5.1.10 Employee Roles ............................................................................. 127
- 5.1.11 View Employee ............................................................................... 127
- 5.1.12 Maintain Titles ............................................................................... 128
- 5.1.13 Add Title ........................................................................................ 128
- 5.1.14 Edit Title ....................................................................................... 129
- 5.1.15 View Title ..................................................................................... 129

#### 6 IMPLEMENTATION GUIDELINES ............................................................. 130

### APPENDICES ............................................................................................... 131

- **A. Schema for common databases** ......................................................... 131
- **B. WSDL for services** ........................................................................... 131
# TABLE OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Interaction of Systems and Security Services</td>
<td>7</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Security Data Structures</td>
<td>8</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Problem with two topics</td>
<td>9</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Single Topic with Service Router</td>
<td>10</td>
</tr>
<tr>
<td>Figure 5</td>
<td>HR and Security System Event Sequence</td>
<td>11</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Legacy Adaptor and Security System Event Sequence</td>
<td>13</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Legacy Adaptor Subscription Process</td>
<td>13</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Legacy Adaptor Update Employee Handler</td>
<td>16</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Legacy Adaptor Update Employee Roles Handler</td>
<td>18</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Legacy Adaptor Get Possible Roles</td>
<td>19</td>
</tr>
<tr>
<td>Figure 11</td>
<td>HR Change Security Router</td>
<td>20</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Handle Employee Change Process</td>
<td>22</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Handle Employee Roles Change Process</td>
<td>23</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Handle Adaptor Subscribe Process</td>
<td>25</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Handle Adaptor Unsubscribe Process</td>
<td>26</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Handle Refresh Roles Timer Event Process</td>
<td>27</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Get Employee Systems Process</td>
<td>31</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Get Employee Roles</td>
<td>32</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Get Possible Systems Process</td>
<td>33</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Get Possible Roles Process</td>
<td>34</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Security Adaptor and Security System Event Sequence</td>
<td>35</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Security Adaptor Change Password Process</td>
<td>40</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Security Adaptor Check Authentication Process</td>
<td>41</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Security Adaptor Authenticate Process</td>
<td>42</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Security Adaptor Check Authorization Process</td>
<td>43</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Security Adaptor Get Authorization Process</td>
<td>44</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Security without Interceptors</td>
<td>45</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Sample Explicit Security Process</td>
<td>46</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Sample Intercepted Security Process</td>
<td>46</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Security with Interceptors</td>
<td>47</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Old Product Database Structure</td>
<td>48</td>
</tr>
<tr>
<td>Figure 32</td>
<td>New Product Database Structure</td>
<td>49</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Product Data Services</td>
<td>50</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Example Employee Portal</td>
<td>54</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Employee Portal Use Cases</td>
<td>55</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Task List Services</td>
<td>57</td>
</tr>
<tr>
<td>Figure 37</td>
<td>Task List Portlet</td>
<td>60</td>
</tr>
<tr>
<td>Figure 38</td>
<td>Add/Update Task Portlet</td>
<td>61</td>
</tr>
<tr>
<td>Figure 39</td>
<td>HR Services</td>
<td>61</td>
</tr>
<tr>
<td>Figure 40</td>
<td>Browse Organization Portlet</td>
<td>65</td>
</tr>
<tr>
<td>Figure 41</td>
<td>Employee List Portlet</td>
<td>65</td>
</tr>
<tr>
<td>Figure 42</td>
<td>Employee View Portlet</td>
<td>65</td>
</tr>
<tr>
<td>Figure 43</td>
<td>Department Listing Portlet</td>
<td>66</td>
</tr>
<tr>
<td>Figure 44</td>
<td>Department View Portlet</td>
<td>66</td>
</tr>
<tr>
<td>Figure 45</td>
<td>Expense Reporting Sequence</td>
<td>67</td>
</tr>
<tr>
<td>Figure 46</td>
<td>Expense Reporting Database</td>
<td>68</td>
</tr>
<tr>
<td>Figure 47</td>
<td>Expense Reporting State Transitions</td>
<td>69</td>
</tr>
<tr>
<td>Figure 48</td>
<td>Expense Reporting Services</td>
<td>70</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Figure 49</td>
<td>Get Reports Service Process</td>
<td>72</td>
</tr>
<tr>
<td>Figure 50</td>
<td>Get Report Service Process</td>
<td>74</td>
</tr>
<tr>
<td>Figure 51</td>
<td>Get Item Types Service Process</td>
<td>75</td>
</tr>
<tr>
<td>Figure 52</td>
<td>Save Report Service Process</td>
<td>77</td>
</tr>
<tr>
<td>Figure 53</td>
<td>Submit Report Service Process</td>
<td>79</td>
</tr>
<tr>
<td>Figure 54</td>
<td>Authorize Report Service Process</td>
<td>81</td>
</tr>
<tr>
<td>Figure 55</td>
<td>Check Cut Service Process</td>
<td>82</td>
</tr>
<tr>
<td>Figure 56</td>
<td>Get Transactions Service Process</td>
<td>84</td>
</tr>
<tr>
<td>Figure 57</td>
<td>Cut Check Service Process</td>
<td>85</td>
</tr>
<tr>
<td>Figure 58</td>
<td>Statefulness of Expense Reporting Services</td>
<td>86</td>
</tr>
<tr>
<td>Figure 59</td>
<td>Expense Reporting Portlet Flow</td>
<td>87</td>
</tr>
<tr>
<td>Figure 60</td>
<td>Portlet and Service Interaction</td>
<td>92</td>
</tr>
<tr>
<td>Figure 61</td>
<td>Employee Review Sequence Phase I</td>
<td>93</td>
</tr>
<tr>
<td>Figure 62</td>
<td>Employee Review Sequence Phase II</td>
<td>94</td>
</tr>
<tr>
<td>Figure 63</td>
<td>Employee Review Database</td>
<td>95</td>
</tr>
<tr>
<td>Figure 64</td>
<td>Employee Review State Transitions</td>
<td>96</td>
</tr>
<tr>
<td>Figure 65</td>
<td>Employee Review Services</td>
<td>97</td>
</tr>
<tr>
<td>Figure 66</td>
<td>Get Reviews Service Process</td>
<td>99</td>
</tr>
<tr>
<td>Figure 67</td>
<td>Get Review Service Process</td>
<td>101</td>
</tr>
<tr>
<td>Figure 68</td>
<td>Save Review Service Process</td>
<td>103</td>
</tr>
<tr>
<td>Figure 69</td>
<td>Submit Review Service Process</td>
<td>105</td>
</tr>
<tr>
<td>Figure 70</td>
<td>Verify Criteria Service Process</td>
<td>107</td>
</tr>
<tr>
<td>Figure 71</td>
<td>Quarter End Timer Process</td>
<td>107</td>
</tr>
<tr>
<td>Figure 72</td>
<td>Rate Review Service Process</td>
<td>109</td>
</tr>
<tr>
<td>Figure 73</td>
<td>Verify Rating Service Process</td>
<td>111</td>
</tr>
<tr>
<td>Figure 74</td>
<td>Process Review Service Process</td>
<td>113</td>
</tr>
<tr>
<td>Figure 75</td>
<td>Update Salary Service Process</td>
<td>114</td>
</tr>
<tr>
<td>Figure 76</td>
<td>Employee Review Portlet Flow</td>
<td>117</td>
</tr>
<tr>
<td>Figure 77</td>
<td>HR Database</td>
<td>121</td>
</tr>
<tr>
<td>Figure 78</td>
<td>HR Web Site Page Flow</td>
<td>122</td>
</tr>
</tbody>
</table>
1 OVERVIEW

The goal of this requirements specification is to provide three major use-cases:

1. Provision of an enterprise wide distributed security mechanism
2. Provision of a product data service
3. Construction of a self-help Employee Portal

Each of these use-cases is then broken down into a number of smaller use-cases, such as Submit Expense Report, Find Employee or Authenticate User. These use-cases are specified in sufficient detail such that they can be implemented unambiguously.

To support these use-cases, additional applications need to be constructed. In general these are CRUD (Create, Read, Update, Delete) applications that support the data structures behind the employee and security services. These applications use component services that are not exposed to other systems.

Each exposed service is defined at the functional level. Where that service invokes other services the required control flow is also explicitly defined both pictorially and in words. Service interfaces are defined using WSDL.

The required backend data structures are defined using Entity Relationship diagrams. It is left to the implementers to determine the best mechanism to utilize these data structures within the services. Similarly, where an artifact requires a User Interface (UI) component, the basic UI layout and contents are defined, but no particular implementation is required.
2  PROVISION OF AN ENTERPRISE WIDE SECURITY SERVICE

2.1 Overview

The SOA based security system provides authentication, authorization and auditing capabilities for all enterprise applications. These applications can be divided into two main categories:

- New Applications
- Legacy Applications

It is assumed that New Applications utilize services directly from the security system to identify users, provide authentication and determine what a user is authorized to do. These services utilize an authentication token that can facilitate a single sign-on (SSO) process for all such applications.

It is further assumed that Legacy Applications, on the other hand, cannot directly utilize these services. Instead, they rely on the security system as a provider of information about new and updated users, their credentials and associated roles. For each legacy system, an adaptor can be built that subscribes to the security system. When security information changes, the security system publishes this information such that the adaptor can modify the local security of the legacy system as required.

The HR system is also integrated with the security system, such that when employees are added or modified the security system is notified and security information is updated appropriately. This interaction between the HR system, new applications, legacy applications and the security system is shown in the following diagram:

Figure 1 - Interaction of Systems and Security Services
This overall system utilizes synchronous services for obtaining security information and asynchronous services for managing and modifying security information. Two different types of asynchronous service are used:

1. Publish / Subscribe Message Broker Services
2. Publish / Subscribe Web Services

The reason for using these two different publish-subscribe mechanisms is based on the simple assumption that the HR system and Security system utilize a common underlying technology platform and can therefore use a platform specific Message Broker, whereas Legacy services may be heterogeneous and require the non-technology specific coupling of web services. For each legacy service, an adaptor needs to be built that complements with the interface specified by the three services. This adaptor is responsible for applying information present in these service calls to the underlying legacy system.

The Message Broker interaction utilizes a publish-subscribe mechanism, rather than a point to point message queue, in order that, in future, other new applications that care about modifications to employees can also subscribe to this service.

### 2.2 Security Data Structures

The security system is supported by the following data structure. Implementations are not restricted to using a database for this information; a directory service such as LDAP could also be utilized.

![Security Data Structures](image)

**Figure 2 - Security Data Structures**
2.3 HR Publishing to Security System

The HR system publishes information about new and updated employees and their roles to the rest of the enterprise through a publish-subscribe message broker system. The Security System is only one of potentially many to subscribe to this, but is the only one to be considered in this specification.

A message broker is used to broadcast these events, rather than direct interaction, so that these changes are guaranteed to be delivered, they can be received by any interested system without modification to the HR system and can happen asynchronously (some might take some time, especially if manual intervention is required).

To achieve this, a single message broker Topic (shorthand terminology for the publishing point or channel of a publish/subscribe mechanisms) – HR change, is used. Subscribers to this topic (in this case the security system) are required to authenticate themselves through credentials supplied by the HR system administrator that are specific to the message broker used at implementation time.

It was initially thought that two topics should be used, one for Employee change events, and one for Role change events. This seems like a natural design pattern, but can lead to sequencing problems, as illustrated by the following diagram:

![Figure 3 - Problem with two topics](image)

If this pattern is employed, after some use of the system, it may be noticed that some Role Change events are not being correctly processed. In particular, at times of high usages (for example when a large number of new employees are being added) some employees may not be assigned roles.

The major assumption here is that adding employees takes much more time than changing roles, since the process waits on legacy systems (one of which may have manual processing involving an email to an administrator who needs to log in to add the user). As a result, there are more items queued on the Employee Change topic than on the role change one. If an additional employee change and associated role change are added to both queues, it is likely that the role change will be processed before the employee change, which may result in failure if the employee needs to be added before the roles are applied.

The problem lies with a lack of sequencing in the processing of asynchronous messages. This is a common issue in a service-oriented system and has a variety of solutions, based on use-case. In some cases, the use of a sequencing or choreography standard, such as BPEL4WS can impose ordering and relationships.
Using such a standard it is possible to define a relationship, or correlation, between the role change and (a related) employee change message and ensure that the processing system waits for an employee change message if it receives a role change message that is correlated to one (role change messages may have no related employee change, in the case of simply updating roles of an existing employee, without modifying employee information).

In this particular case, however, a simpler solution is possible – merge the two topics. The HR system should publish all employee change and roles change messages on a single topic – HR Change. Within the security system, a process subscribes to this single topic. On receipt of messages, a service router is used to invoke either the employeeChange or rolesChange process, depending on the message received.

![Figure 4 - Single Topic with Service Router](image)

This will ensure that the messages that have a serial dependency are processed in sequence, since a roles change message will be queued behind an associated employee change message. The messages being exchanged have not changed, only the transmission pattern.

The ordering of events between the two systems can be seen in the following sequence diagram:
The following sections define each of the published messages. The employeeChange and rolesChange services are detailed later in this specification.

### 2.3.1 The HR Change Topic

Within the message broker there needs to be a Topic defined (that is a broadcast destination to which interested parties can subscribe and upon which messages are published) with the name of HR Change. The HR System will publish messages to this Topic whenever an employee is added to the system or modified in any way (including being deleted), or when employee roles are modified.

Subscription to the hrChange Topic is through the message broker specific subscription mechanism. It is a requirement that this subscription is secure and utilizes credentials specific to the system subscribing. Definition and use of these credentials depends on the implementation and is outside the scope of this specification.

### 2.3.2 Employee Change message

The employeeChange message is published when an employee is added, modified or deleted and has the following signature:

**Type:** asynchronous

**Client sends:** employeeChange

**Contents:** EmployeeDetails

- **ID** (integer 1)
- **Operation** (string 1)
  - Added
  - or Updated
  - or Deleted
- **FirstName** (string 1)
- **MiddleName** (string 0-1)
- **LastName** (string 1)
- **SSN** (string 1)
- **DateOfBirth** (date 1)
- **Address** (complex 1)
  - **StreetAddress** (string 1)
  - **City** (string 1)
  - **State** (string 1)
  - **Zip** (string 1)
- **ContactInfo** (complex 1)
  - **Email** (string 1)
  - **BusinessPhone** (string 1)
  - **HomePhone** (string 0-1)
  - **MobilePhone** (string 0-1)
- **RequestedSystems** (complex 1)
  - **System** (complex 0-*)
    - **ID** (integer 1)
    - **Operation** (string 1)
      - Add
      - or Remove
- **Security:** authCode + signing + encryption
- **Protocol:** Message Broker

**Client receives:** none
Patterns Covered:

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>employeeChange is an asynchronous message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish-Subscribe Services</td>
<td>message is published on the hrChange topic managed by the HR system</td>
</tr>
</tbody>
</table>

2.3.3 Roles Change message

The rolesChange message is published when the roles of an employee have been changed and has the following signature:

**Type:** asynchronous

**Client sends:** rolesChange

**Contents:** RolesDetails

- Employee
  - ID
- Systems
  - System
    - ID
- SystemRoles
  - Role
    - ID
  - SystemRoles
    - Role
      - ID
  - SystemRoles
    - Role
      - ID
- SystemRoles
  - Role
    - ID
  - SystemRoles
    - Role
      - ID
- Operation
  - Granted
  - or Revoked
  - #enum

**Security:** authCode + signing + encryption

**Protocol:** Message Broker

**Client receives:** none

Patterns Covered:

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>rolesChange is an asynchronous message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish-Subscribe Services</td>
<td>message is published on the hrChange topic managed by the HR system</td>
</tr>
</tbody>
</table>

2.4 The Legacy Adaptor

For each Legacy system (here we mean a system that cannot use the security systems authentication and authorization services for some reason) the legacy adaptor plays three roles:

1. Implements the getPossibleRoles service to return a list of roles valid on that system to the security system to enable employees to be assigned roles by an HR representative
2. Subscribes to the updateEmployee service and handles adding / removing / updating login information for that employee whenever such a message is published
3. Subscribes to the updateEmployeeRoles service and handles granting or revoking roles for that employee whenever such a message is published
The actual implementation of the adaptor may be very different for each system, but the
overriding requirement is that each adaptor must implement each one of these services. The
subscription mechanism is different from the HR system, using Web services to subscribe,
unsubscribe and in the publishing of messages. This is shown in the following sequence:

Figure 6 - Legacy Adaptor and Security System Event Sequence

2.4.1 Subscribe to the Security System

Subscription to the Security System by a legacy adapter is a two phase process. Since it is not
possible to determine the credentials of a particular system, and confidential information will be
published to subscribers, an additional layer of trust is required. The recommended approach is
for the legacy adaptor to initiate subscription via a synchronous subscribe message that returns
a unique system id. A public/private key pair is then generated and the public key is given to
the security administrator by the legacy administrator.

Figure 7 - Legacy Adaptor Subscription Process
Once the manual process is complete, the legacy system is fully subscribed. Messages sent between the security system and legacy adaptor should be encrypted and decrypted by using the generated key pair.

The adaptor subscribes to the security service through a synchronous subscribe message that provides the security system with service access point information for the adaptor’s implementation and receives back a unique system ID to use in future getPossibleRoles and unsubscribe messages. It has the following signature:

**Type:** synchronous

**Service sends:** subscribe

**Contents:** Subscription

<table>
<thead>
<tr>
<th>Part</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Complex</td>
<td>1</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>0-1</td>
</tr>
<tr>
<td>AccessPoints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getPossibleRoles</td>
<td>Complex</td>
<td>1</td>
</tr>
<tr>
<td>updateEmployee</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>updateEmployeeRoles</td>
<td>String</td>
<td>1</td>
</tr>
</tbody>
</table>

**Security:** https + signing

**Protocol:** http

**Service receives:** subscribeResponse

**Contents:** SubscriptionInfo

<table>
<thead>
<tr>
<th>Part</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Complex</td>
<td>1</td>
</tr>
<tr>
<td>ID</td>
<td>Integer</td>
<td>1</td>
</tr>
</tbody>
</table>

or: signatureIncorrectFault

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Service Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Services</td>
<td>subscribe is a synchronous message</td>
</tr>
<tr>
<td>Publish-Subscribe Services</td>
<td>subscribe adds the specified system into the list of adaptors to which messages will be published</td>
</tr>
</tbody>
</table>

The Subscribe message is handled by the security system and the process for the handling of this message is described in 2.5.1.6.

### 2.4.2 Unsubscribe from the Security System

The adaptor unsubscribes from the security service through a synchronous unsubscribe message, which has the following signature:

**Type:** synchronous

**Service sends:** unsubscribe

**Contents:** Subscription

<table>
<thead>
<tr>
<th>Part</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Complex</td>
<td>1</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>ID</td>
<td>Integer</td>
<td>1</td>
</tr>
</tbody>
</table>

**Security:** https + signing

**Protocol:** http
Service receives: unsubscribeResponse
Contents: none

or: signatureIncorrectFault

Patterns Covered:

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>subscribe is a synchronous message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish-Subscribe Services</td>
<td>subscribe adds the specified system into the list of adaptors to which messages will be published</td>
</tr>
</tbody>
</table>

The Unsubscribe message is handled by the security system and the process for the handling of this message is described in 2.5.1.7

2.4.3 Handle Update Employee

The updateEmployee is received by an adaptor that has subscribed to the security service when changes to an employee have been made. The message has the following signature:

Type: asynchronous

Client sends: updateEmployee
Contents: EmployeeDetails
    ID: Integer 1
    SystemUserName: String 0-1
    Operation: String 1
        Added #enum
        or Updated #enum
        or Deleted #enum
    FirstName: String 1
    MiddleName: String 0-1
    LastName: String 1
    SSN: String 1
    DateOfBirth: Date 1
    Address: Complex 1
        StreetAddress: String 1
        City: String 1
        State: String 1
        Zip: String 1
    ContactInfo: Complex 1
        Email: String 1
        BusinessPhone: String 1
        HomePhone: String 0-1
        MobilePhone: String 0-1
    Security: http + signing + encryption
    Protocol: http

Client receives: none

Patterns Covered:

| Asynchronous Services | updateEmployee is an asynchronous |
The actual implementation of this process will differ per adaptor. It may be an automated process (such as running a database script, or executing some APIs on an ERP system) or it might require manual intervention, in which case some admin would be notified by email that they must add a user, followed by clicking on a link when they had done so, which would trigger the sending of the confirmation message. There are, however, two requirements:

1. The employee should be notified that they have been added to the specified system via email and this email should include their user name and default password for that system.
2. The security system should be notified when the process is complete, whether it is successful or not.

If the process succeeds, the completed operation and system user name (if the operation is to add a user) should be returned in the confirmation message. If the process fails, a message describing the reason for failure should be returned within the confirmation message.

The contents and processing for this confirmation message are described in a later section.

The process executed for this service is outlined in the following diagram:
2.4.4 Handle Update Employee Roles

The updateEmployeeRoles message has the following signature:

**Type:** asynchronous

**Client sends:** updateEmployeeRoles

**Contents:** RolesDetails

<table>
<thead>
<tr>
<th>Role</th>
<th>ID</th>
<th>Name</th>
<th>Operation</th>
<th>Granted or Revoked</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#enum</td>
</tr>
</tbody>
</table>

Security: http + signing + encryption

**Client receives:** none

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>updateEmployeeRoles is an asynchronous message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish-Subscribe Services</td>
<td>message is published by the security system to all subscribing adaptors</td>
</tr>
</tbody>
</table>

Again, the actual implementation of this process will differ per adaptor.

It is required that the security system should be notified when the process is complete, whether it is successful or not. The confirmation message should provide details about the role’s status. The contents and processing for this confirmation message are described in a later section.

The process executed for this service is outlined in the following diagram:
Figure 9 - Legacy Adaptor Update Employee Roles Handler

may be a manual process and take some time

For Each Role

operation = granted
operation = revoked

update employee permissions to include
generated role

update employee permissions to remove revoked role

send updateEmployee Confirmation
2.4.5 Get Possible Roles

The getPossibleRoles message has the following signature:

**Type:** synchronous

**Client sends:**
- *getPossibleRoles*
  - *Contents:* none
  - *Security:* http
  - *Protocol:* http

**Client receives:**
- *listOfPossibleRoles*
  - *Contents:* PossibleRoles
    - *System*
    - *ID*
    - *Roles*
      - *Role*
        - *LocalID*
        - *Name*
        - *Description*

Patterns Covered:

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>updateEmployeeRoles is an asynchronous message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish-Subscribe Services</td>
<td>message is published by the security system to all subscribing adaptors</td>
</tr>
</tbody>
</table>

The process executed for this service is outlined in the following diagram:

![Figure 10 - Legacy Adaptor Get Possible Roles](image)
2.5 Security System Services

The security system itself provides three sets of services:

1. **User Propagation Services**
   - The HR System and Legacy Adaptors integrate with these services as mentioned previously. They include the listeners for employee and roles change and the publishing to subscribed adaptors.

2. **User Information Services**
   - These services are used by the HR system to obtain information about current systems and roles for a given employee as well as the possible ones.

3. **Security Adaptor Services**
   - New Applications integrate with the security system through these services for authentication and authorization purposes.

These following sections explain these services in detail.

2.5.1 User Propagation Services

2.5.1.1 Service Router for HR Change

The service router subscribes to the HR Change Topic and is invoked on the receipt of employeeChange or rolesChange messages on this topic. The process executed is shown in the following diagram:

![HR Change Security Router Diagram](image-url)
2.5.1.2 Handle Employee Change Message

The security service to handle employee changes is invoked by the service router on the receipt of an employeeChange message as defined in 2.3.2. The process executed is shown in the Figure 12 - Handle Employee Change Process.

2.5.1.3 Handle Roles Change Message

The security service to handle roles change is invoked by the service router on the receipt of a rolesChange message as defined in 2.3.3. The process executed is shown Figure 13 - Handle Employee Roles Change Process.

2.5.1.4 Update Employee Confirmation Message

The updateEmployeeConfirmation message is sent by an adaptor that has subscribed to the security service it has made or failed to make changes as specified in the updateEmployee message sent from the employeeChange handler. The message has the following signature:

<table>
<thead>
<tr>
<th>Type:</th>
<th>asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service sends:</td>
<td>updateEmployeeConfirmation</td>
</tr>
<tr>
<td>Contents:</td>
<td>EmployeeDetails</td>
</tr>
<tr>
<td>ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>SystemUserName</td>
<td>String 1</td>
</tr>
<tr>
<td>SystemID</td>
<td>String 1</td>
</tr>
<tr>
<td>Operation</td>
<td>String 1</td>
</tr>
<tr>
<td>Added</td>
<td>#enum</td>
</tr>
<tr>
<td>or Updated</td>
<td>#enum</td>
</tr>
<tr>
<td>or Deleted</td>
<td>#enum</td>
</tr>
<tr>
<td>Status</td>
<td>String 1</td>
</tr>
<tr>
<td>FailureMessage</td>
<td>String 0-1</td>
</tr>
</tbody>
</table>

| Security: | http + signing |
| Protocol: | http |

Service receives: none

Patterns Covered:

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>updateEmployeeConfirmation is an asynchronous message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish-Subscribe Services</td>
<td>message is returned all subscribing adaptors after completion of the process to handle the updateEmployeeMessage</td>
</tr>
</tbody>
</table>
Figure 12 - Handle Employee Change Process
2.5.1.5 Update Employee Roles Confirmation Message

The updateEmployeeRolesConfirmation message is sent by an adaptor that has subscribed to the security service it has made or failed to make changes as specified in the updateEmployeeRoles message sent from the rolesChange handler. The message has the following signature:

**Type:** asynchronous

**Service sends:** updateEmployeeRolesConfirmation

**Contents:** RolesDetails

- SystemID
- Employee
- ID
- SystemUserName
- Roles
- Role
- ID
- SystemRoleID

- Complex 1
- String 1
- Complex 1
- Integer 1
- String 1
- Complex 1
- Complex 0-*
- Integer 1
- String 1
2.5.1.6 Handle Adaptor Subscribe

The security service to handle adaptor subscribe messages has a signature as defined in 2.4.1. The process executed is shown in Figure 14 - Handle Adaptor Subscribe Process.

2.5.1.7 Handle Adaptor Unsubscribe

The security service to handle adaptor unsubscribe messages has a signature as defined in 2.4.2. The process executed is Figure 15 - Handle Adaptor Unsubscribe Process

2.5.1.8 Handle Refresh Roles Timer Event

The refreshRoles timer is used to schedule updates of possible roles for each subscribed system. A timer is created for each adaptor and carries the SystemID. When an adaptor first subscribes to the security system, a timer is started to run immediately. Once this first event is handled the timer is reset to run once a week. If a system unsubscribed, the timer is stopped. The Timer Event process is shown in Figure 16 - Handle Refresh Roles Timer Event Process.
Figure 14 - Handle Adaptor Subscribe Process
Figure 15 - Handle Adaptor Unsubscribe Process
Figure 16 - Handle Refresh Roles Timer Event Process
2.5.2 User Information Services

2.5.2.1 Get Employee Systems

The getEmployeeSystems message has the following signature:

Type: synchronous

Client sends: getEmployeeSystems
  Contents: Employee
    ID
  Security: auth token
  Protocol: http

Client receives: employeeSystems
  Contents: EmployeeSystemsDetail
    Employee
      ID
    Systems
      System
        ID
        Name
        Description
  ID

or: authFault

Patterns Covered:

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>getEmployeeSystems service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is outlined Figure 17 - Get Employee Systems Process

2.5.2.2 Get Employee Roles

The getEmployeeRoles message has the following signature:

Type: synchronous

Client sends: getEmployeeRoles
  Contents: Employee
    ID
  Security: auth token
  Protocol: http

Client receives: employeeRoles
  Contents: EmployeeRolesDetail

or: authFault
Patterns Covered:

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>getEmployeeRoles service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is outlined in Figure 18 - Get Employee Roles

2.5.2.3 Get Possible Systems

The getPossibleSystems message has the following signature:

Type: synchronous

Client sends: getPossibleSystems
- Contents: none
- Security: auth token
- Protocol: http

Client receives: possibleSystems
- Contents: PossibleSystemsDetail
  - Systems
  - System (0-*)
  - ID
  - Name
  - Description

or: authFault

Patterns Covered:

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>getPossibleSystems service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>
The process executed for this service is outlined in Figure 19 - Get Possible Systems Process

### 2.5.2.4 Get Possible Roles

The getPossibleRoles message has the following signature:

**Type:** synchronous

**Client sends:** getPossibleRoles

*Contents:*

- Employee: Complex 1
  - ID: Integer 1

*Security:* auth token

*Protocol:* http

**Client receives:** possibleRoles

*Contents:*

- PossibleRolesDetail: Complex 1
  - Employee: Complex 1
    - ID: Integer 1
  - Systems: Complex 1
    - System: Complex 0-*
      - ID: Integer 1
      - Name: String 1
      - Description: String 0-1
  - SystemRoles: Complex 1
    - Role: Complex 0-*
      - ID: Integer 1
      - Name: String 1
      - Description: String 0-1

*or:* authFault

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>getPossibleRoles service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is outlined in Figure 20 - Get Possible Roles Process
Figure 17 - Get Employee Systems Process
Figure 18 - Get Employee Roles
Figure 19 - Get Possible Systems Process
Figure 20 - Get Possible Roles Process
2.5.3 Security Adaptor Services

The Security Adaptor Services do not constitute a security adaptor in themselves, rather they are the services required by an adaptor created for a specific environment. They are not limited to a single implementation, but an adaptor using these services is required to manage their invocation. Additionally, an adaptor may have its own data store for auth tokens, roles, etc. The services may be used to synchronize the adaptors data with that of the centralized service. It may additionally subscribe to employee and role change messages in the same way as a legacy adaptor to trigger synchronization. This may be necessary to ensure satisfactory performance for a high load system.

For example, a typical adaptor will be able to intercept page requests. If pages have been marked as secure, the adaptor should first check if an auth token is present and that it is valid. This is done through the checkAuthenticated service.

If a user needs to be authenticated (through a login-page for example) the authenticate service should be called. Once authenticated the checkAuthorization or getAuthorization services can be called to find out what roles the user has been assigned. The adaptor can use these roles to determine if certain resources should be accessed.

The change password service is not, strictly speaking part of the adaptor, since it requires a UI implementation to complete it, however it is provided for use by applications running on the environment managed by the security adaptor.

The overall sequence of events between a user, a security adaptor implementation and the security system is shown in the following diagram:
2.5.3.1 Change Password

Change password may be called by any application and is accessed by an authenticated (logged in) user. An application facility applications is required to allow the entry of the new password, old password, a password question and a corresponding answer (for retrieval of forgotten passwords). The changePassword message has the following signature:

**Type:** synchronous

**Client sends:** changePassword

**Contents:** PasswordInfo

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserID</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>OldPassword</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>NewPassword</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>PasswordQuestion</td>
<td>String</td>
<td>0-1</td>
</tr>
<tr>
<td>PasswordAnswer</td>
<td>String</td>
<td>0-1</td>
</tr>
</tbody>
</table>

**Security:** https + auth token

**Protocol:** https

**Client receives:** changePasswordResults

**Contents:** PasswordChangeDetails

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserID</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>ChangeStatus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Success</td>
<td>String</td>
<td>#enum</td>
</tr>
<tr>
<td>Failure</td>
<td>String</td>
<td>#enum</td>
</tr>
<tr>
<td>FailureReason</td>
<td></td>
<td>String 0-1</td>
</tr>
</tbody>
</table>

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>changePassword service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is outlined in Figure 22 - Security Adaptor Change Password Process.

2.5.3.2 Check Authenticated

Check authenticated may be called by any application to verify that an authentication token is valid for a particular user. The token is returned from the authenticate service and should be stored by some mechanism for a particular browser session. Applications should use this service to protect any secure page. The checkAuthenticated message has the following signature:

**Type:** synchronous
Client sends: checkAuthenticated

Contents: AuthenticationInfo Complex 1
          AuthToken String 1
          UserID Integer 1

Security: http
Protocol: http

Client receives: checkAuthenticatedResults

Contents: AuthenticationDetails Complex 1
          AuthenticationStatus String 1
          Success #enum
          or Failure #enum

Patterns Covered:

<table>
<thead>
<tr>
<th>Patterns Covered</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Services</td>
<td>checkAuthenticated service is synchronous</td>
</tr>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is outlined in Figure 23 - Security Adaptor Check Authentication Process.

2.5.3.3 Authenticate

Any application requiring security may provide a log in capability for the user. The authenticate service can be called by any application to verify the identity of a user. The authenticate message has the following signature:

Type: synchronous

Client sends: authenticate

Contents: AuthenticationInfo Complex 1
          UserName String 1
          Password String 1

Security: https + encryption
Protocol: https

Client receives: authenticateResults

Contents: AuthenticationDetails Complex 1
          UserID Integer 1
          AuthToken String 0-1
          or FailureReason String 0-1

Patterns Covered:

<table>
<thead>
<tr>
<th>Patterns Covered</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Services</td>
<td>authenticate service is synchronous</td>
</tr>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control</td>
</tr>
</tbody>
</table>
The process executed for this service is outlined in Figure 24 - Security Adaptor Authenticate Process.

### 2.5.3.4 Check Authorization

Once a user is authenticated, any application may call check authorization to see if a user has been granted a specific role. This enables applications to customize interfaces and secure functions based on the user’s roles. The checkAuthorization message has the following signature:

**Type:** synchronous

**Client sends:** checkAuthorization

**Contents:** AuthorizationInfo

- **UserID** (Integer 1)
- **Role** (Complex 1)
- **ID** (Integer 1)

**Security:** http + auth token

**Protocol:** http

**Client receives:** checkAuthorizationResults

**Contents:** AuthorizationDetails

- **AuthorizationStatus** (String 1)
- **Success** (#enum)
- **Failure** (#enum)

or: authFault

#### Patterns Covered:

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>checkAuthorization service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is outlined in Figure 25 - Security Adaptor Check Authorization Process.

### 2.5.3.5 Get Authorization

Once a user is authenticated, any application may call get authorization to obtain a list of roles that have been granted to that user. This enables applications to customize interfaces and secure functions based on the user’s roles. The getAuthorization message has the following signature:

**Type:** synchronous

**Client sends:** getAuthorization

**Contents:** AuthorizationInfo

- **Complex** 1
UserID: Integer 1

Security: http + auth token  
Protocol: http

Client receives: `getAuthorizationResults`  
Contents: `AuthorizationDetails`  
Roles: Complex 1  
  Role: Complex 0-*  
  Name: String 1  
  ID: Integer 1

or: `authFault`

Patterns Covered:

<table>
<thead>
<tr>
<th>Patterns Covered</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Services</td>
<td>getAuthorization service is synchronous</td>
</tr>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is outlined in Figure 26 - Security Adaptor Get Authorization Process.
Figure 22 - Security Adaptor Change Password Process
Figure 23 - Security Adaptor Check Authentication Process
Figure 24 - Security Adaptor Authenticate Process
Figure 25 - Security Adaptor Check Authorization Process
Figure 26 - Security Adaptor Get Authorization Process
2.6 Provision of Security Services through Intercepts

Currently, security services are explicitly included in many processes within this specification as steps in the process. There are two main security services currently in use:

- Check Authentication
- Check Signature

There are many other potential services that could be added, most notable encryption / decryption, logging and auditing. Adding additional services to each process, complete with failure handling could lead to unwieldy and complex process definitions. The recommended approach here is to take a page out of the Aspect Oriented Programming (AOP) book. Each of these features is a “cross-cutting concern” – an aspect of many processes that needs to be applied in a consistent way.

The pattern used to achieve this in SOA is an intercept. As the name suggests it sits between the service consumer and the provider, intercepting the message and taking some action upon it. In the way, the service provider can simply concentrate on the business purpose of their process. Security services are added during administration – as a rule, developers should not need to code security into the application.

This does not change the security requirements of processes, but changes the way these requirements are added to the process. The following diagram shows how processes engage security services without intercepts:

![Diagram of Security without Intercepts]

Figure 27 - Security without Intercepts
For every service provided by the security system (this is not restricted to message based services, utilities that sign a message, or deal with encryption are also services) there is a corresponding potential node within a process. Each node needs to be included explicitly in the process at the correct point, and failure of the service and corresponding fault response needs to be encoded specifically. This leads to a process design, and resulting implementation that is difficult to follow and hard to maintain for many processes.

The following diagram shows the same process with and without security interception:

![Sample Explicit Security Process](Figure 28)

![Sample Intercepted Security Process](Figure 29)

The process definition would grow if more services were included, the developer of this process needs to be concerned about security, and every process would need to be changed if security requirements changed.
By introducing security through interceptors, the model changes. The security aspects of an application are applied by an administrator by setting properties. For example, a process can be flag such that it requires authentication, the message is signed and encrypted and access should be logged. The following diagram shows how security is simplified through interceptors.

Figure 30 - Security with Interceptors

Note that the message is intercepted at both the consumer (send) and provider (receive) ends of the interaction. At the consumer end, a service invocation can be marked as requiring authentication (specifying the credentials to use), needing to be signed or encrypted or that the invocation needs to be logged. At the provider end, the service can be marked as requiring authentication, that it should be signed, it requires decryption or that use of the service should be logged.

In this way, processes become independent of security. No additional explicit process nodes are required and fault handling can be done outside of the process and in a standard way across all provided services.

Although the specification explicitly defines some aspects of security within the process model diagrams, these should ideally be treated as intercepted aspects in a final implementation.
3 PROVISION OF A PRODUCT DATA SERVICE

3.1 Overview

Like many enterprises, our example enterprise has grown over time. As a result, data pertaining to many similar entities is present in many data sources. One particular case, in this specification, is that of product information. There are two data sources containing information about products. They are both relational databases, but have very different schema.

The older database contains only basic product information and was in place before the Web became a way of distributing product catalogs. The newer database was built to contain more information about products, to include web images and handle change management.

The Product Data Service provides a single point of entry into both data sources. This allows systems to obtain product information from both places using a single query mechanism. The data service also adds defaulted information into the returned data structures when that information is not present in the old system.

3.2 Data Structures

The older database has the following structure:

![Old Product Database Structure](image)

This represents a tree of categories, each category having associated product information. Each product belongs to a category and has associated with it keywords present in a single column entry. Additional information was added to the product in the form of an item that included its price, cost to make and status that determined its availability. This was then further extended by adding additional inventory information in an additional table.

The new product database was part of an eCommerce initiative, but for legacy reasons older products cannot be migrated into it. This database has the following structure:
The new database has a slightly different structure and contains much more information. There is still a tree of categories, but now products can belong to more than one category by way of the product/category many to many table. Product keywords are now stored in a separate table to facilitate faster searching. The product and category tables contain change management information about the users that created and modified them and also image and page information used to display these products on the Web. Inventory information is not present in this database as it now belongs to the Supply Chain system and can be accessed through a service call.

Figure 32 - New Product Database Structure
3.3 Services

The product data service provides four services to query returned lists of products or categories and detailed information about single products or categories. These are shown in the following diagram:

![Product Data Services Diagram]

These services may be invoked freely by any corporate portal (Employee, Sales, or Customer) or any other corporate system, providing maximum reuse. For the purpose of this specification, however, a particular consumer is not defined.

3.3.1 Get Categories

The getCategory service returns a list of categories based on query criteria. Categories can be queried based on a parent id, product id or partial name. It has the following signature:

**Type:** synchronous

**Client sends:** getCategory

**Contents:** CategoryQuery

- `ParentCategoryID`
- `ProductID`
- `NameContains`

**Security:** none

**Protocol:** http

**Client receives:** categoriesResults
Since the two databases have different category id structures and the old database does not contain image information, the service will have to perform the following tasks:

1. Generate a unique Category/ID between databases. This could be something like OLD-{CATID} and NEW-{CATEGORY_ID}. This id should be used in future queries as the parent id or category id in product searches.
2. Retrieve categories based on a product id that has been generated in product searches using a similar mechanism
3. Retrieve Name and Short Description from both databases
4. Retrieve image information from the new database, but generate image URLs and alt text based on category name for the old database

### 3.3.2 Get Category

The getCategory service returns the details of a single category based on the unique generated category id. The response contains more detailed category information, as well as references to the parent category, if any, and any child categories. It has the following signature:

**Type:** synchronous

**Client sends:** getCategory
- **Contents:** Category
  - ID
  - String
  - Name
  - ShortDescription
  - LongDescription
  - LargeImageUrl
  - LargeImageAltText
  - ParentCategory
    - ID
  - Name
  - ShortDescription
  - SmalImageUrl
  - SmalImageAltText
  - ChildCategories
    - Category
      - ID
      - String
      - Name
      - ShortDescription
      - SmallImageUrl
      - SmallImageAltText

**Service receives:** categoryResult
- **Contents:** Category
  - ID
  - String
  - Name
  - ShortDescription
  - LargeImageUrl
  - LargeImageAltText
  - ParentCategory
    - ID
    - String
    - Name
    - ShortDescription
    - SmallImageUrl
    - SmallImageAltText
  - ChildCategories
    - Category
      - ID
      - String
      - Name
      - ShortDescription
      - SmallImageUrl
      - SmallImageAltText
As with the getCategories service, this service also needs to deal with aggregation and creation of default data for image URLs. The ID passed into this service is of the format of the unique category id as specified in the previous section.

### 3.3.3 Get Products

The getProducts service returns a list of products based on query criteria. Products can be queried based on any combination of category id, product name, keywords, and minimum / maximum price. This service has the following signature:

**Type:** synchronous

**Client sends:** getProducts

**Contents:** ProductQuery Complex 1

- CategoryID String 0-1
- NameContains String 0-1
- Keywords String 0-1
- Keyword String 0-*
- Price Complex 0-1
  - Minimum Decimal 0-1
  - Maximum Decimal 0-1

**Security:** none

**Protocol:** http

**Client receives:** productsResults

**Contents:** ListOfProducts Complex 1

- Product Complex 0-*
  - ID String 1
  - Name String 1
  - ShortDescription String 0-1
  - SmallImageUrl String 1
  - SmallImageAltText String 1
  - Price Decimal 1

Since the two databases have different product id structures and the old database does not contain image information, the service will have to perform the following tasks:

1. Generate a unique Product/ID between databases. This could be something like OLD-{PRODUCTID} and NEW-{SKU}. This id should be used in future queries to get product details or retrieve a list of categories that product belongs to.
2. Retrieve products based on a category id that has been generated in category searches using a similar mechanism
3. Retrieve Name and Short Description from both databases
4. Retrieve image information from the new database, but generate image URLs and alt text based on category name for the old database

### 3.3.4 Get Product

The getProduct service returns details about a particular product based on the unique generated product id. This service has the following signature:

**Type:** synchronous

**Client sends:** getProduct

**Contents:** Product Complex 1
As with the getProducts service, this service also needs to deal with aggregation and creation of default data for image URLs. The ID passed into this service is of the format of the unique product id as specified in the previous section.

The Inventory for old products is retrieved directly from the inventory table. For new products, a service provided by the Supply Chain System is used to obtain inventory.

This service has the following signature:

**Type:** synchronous

**Client sends:**

<table>
<thead>
<tr>
<th>Contents:</th>
<th>getInventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security:</td>
<td>http</td>
</tr>
<tr>
<td>Protocol:</td>
<td>http</td>
</tr>
</tbody>
</table>

**Client receives:**

<table>
<thead>
<tr>
<th>Contents:</th>
<th>inventoryResult</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKU</td>
<td>Product</td>
</tr>
<tr>
<td>Inventory</td>
<td></td>
</tr>
<tr>
<td>EarliestShipDate</td>
<td>Date 0-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SKU</th>
<th>String 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory</td>
<td>Integer 1</td>
</tr>
<tr>
<td>ID</td>
<td>String 1</td>
</tr>
<tr>
<td>Name</td>
<td>String 1</td>
</tr>
<tr>
<td>Price</td>
<td>Decimal 1</td>
</tr>
<tr>
<td>Keywords</td>
<td>Complex 1</td>
</tr>
<tr>
<td>Category</td>
<td>Complex 0-*</td>
</tr>
<tr>
<td>ID</td>
<td>String 1</td>
</tr>
<tr>
<td>Name</td>
<td>String 1</td>
</tr>
<tr>
<td>Inventory</td>
<td>Integer 1</td>
</tr>
<tr>
<td>ShippingDate</td>
<td>Date 1</td>
</tr>
</tbody>
</table>
4 CONSTRUCTION OF AN EMPLOYEE PORTAL

4.1 Overview

The Employee Portal is an example of a Service Consumer. It is a Web site that enables an employee to carry out a number of operations supporting their job. Each operation is backed by services. It is a requirement that the Employee Portal does not have direct access to any of the data stores providing information or supporting operations the employee can carry out.

Instead, the portal is comprised of several self-contained portlets each of which retrieve information and carry out operations through back-end services based on user interaction.

![Employee Portal](image)

Figure 34 - Example Employee Portal

Figure 34 shows an example Employee Portal. The Self-Help section is shown for completeness, but is outside the scope of this specification. The portions of the portal defined in this specification are:

1. Login
2. Task List
3. Employee Lookup
4. Expense Report Process
5. Employee Review process

Each of these use-cases is made up of a number of operations that are provided by services. The operations somewhat mirror the interaction the user will have with the portal.
These are summarized in the following Use Case diagram:

![Use Case Diagram](image)

Figure 35 - Employee Portal Use Cases

Some of these operations (such as Get Departments, Get Transactions or Find Employee) are business services. That is they are self contained atomic services that do not require additional processing and do not change the state of some item. Other operations (such as Submit Report or Process Review) are workflow services that are part of a long-lived transaction carrying state information and additional backend processing.

The following sections detail each of the use-cases and the operations that they contain.

### 4.2 Authenticate

The Authentication mechanism for the Employee Portal should utilize the services provided by a security adaptor configured for the portal as defined in section 2.5.3. At the minimum this should provide four capabilities:

1. Checking of authentication for all access requiring login before access is allowed
2. Provision of login / logout functionality
3. Provision of a change password mechanism
4. Utilization of users roles to determine access to particular functionality

The security adaptor implementation for the portal will depend on inherent security features available on the portal platform. Additional customization features such as saving a password, or personalization may be incorporated but are outside the scope of this specification.
4.2.1 Check Authentication

Whenever a user attempts to access the Employee Portal or one of its internal pages the Portal should check to see if the user is authenticated. Some mechanism should be provided to access an authentication token from the user's session (corresponding to a browser session) and check the authentication of this, if present, through the checkAuthenticated security service (see 2.5.3.2). If the token is not present or incorrect, the user should be forced to login before any further access is possible.

4.2.2 Login / Logout

The login mechanism should utilize existing Portal functionality where possible. If a predefined Login portlet exists, this should be used. Processing of the login itself should be delegated to the authenticate (see 2.5.3.3) security service with appropriate action taken on failure.

The portlet provided should have the following basic outline:

![Login Portlet Diagram]

If the security service fails the failureReason should be displayed, if it succeeds the login portlet should be replaced by the task list for that user (see 4.3). Once authenticated a change password (see 4.2.3) link and logout link should be provided. Logout should simply clear the authentication token for the user and end the session such that upon reloading the portal they will once again be required to login.

4.2.3 Change Password

Once logged in the user should be able to modify their password through a change password portlet. If an existing portlet exists, this can be used providing it matches the required data contents in the outline. Processing of the change of password should be delegate to the changePassword (see 2.5.3.1) security service with appropriate action taken on failure.

The portlet provided should have the following basic outline:

![Change Password Portlet Diagram]

As with login, if the security service fails, the failure reason should be displayed.
4.2.4 Role Based Access Control

Certain aspects of the portal should utilize the authorization capabilities of the security adaptor. This could be achieved through checkAuthorization (see 2.5.3.4) or getAuthorization (see 2.5.3.5) depending on the implementation of the portal’s security functionality. The roles assigned to a user should be used to control access to certain resources. For example, the Expense Reporting portlet will only show Staff Reports if the user has the role of a manager, or accountant; only those users with the HR role should be able to access the HR application; etc.

4.3 Manage Task List

All users logged into the employee portal have access to their task list. This provides a mechanism for organizing and communicating items that employees need to take some action on. There are two major types of tasks:

1. Automated tasks
2. User defined tasks

In this specification there are two kinds of automated tasks, one for expense reporting, and the other for employee reviews. These tasks will appear in the list of the employee currently required to work on them. Expense report and employee review tasks are assigned to particular users (employee, manager, accountant, hr rep), as their state changes, through the services invoked to work with them. In addition to these, users may add and manage their own tasks for other items they wish to appear in the list.

The interaction of the task list portlet, task management services and other portlets are shown in the following diagram:

![Figure 36 - Task List Services]

The following sections define the interfaces to these services in detail. Due to the many ways in which task management could be performed within a particular implementation no specific requirements about data structures or service implementations are included.
4.3.1 Add Task

This service adds a task to the employee’s task list. The service signature is as follows:

**Type:** synchronous

**Client sends:**
- `addTask`
  - **Contents:**
    - `TaskDetails` (Complex 1)
      - `Employee` (Complex 1)
      - `ID` (Integer 1)
      - `Task` (Complex 1)
      - `Subject` (String 1)
      - `DueDate` (Date 0-1)
      - `Status` (String)
        - NotStarted #enum
        - In Progress #enum
        - Completed #enum
      - `Description` (String 0-1)
  - **Security:** auth token
  - **Protocol:** http

**Client receives:**
- `addTaskResult`
  - **Contents:**
    - `TaskDetails` (Complex 1)
      - `Task` (Complex 1)
      - `ID` (Integer 1)
  - **or:** `authFault`
  - **or:** `addTaskFault`

4.3.2 Update Task

This service updates a task from an employee’s task list. The service signature is as follows:

**Type:** synchronous

**Client sends:**
- `updateTask`
  - **Contents:**
    - `TaskDetails` (Complex 1)
      - `Employee` (Complex 1)
      - `ID` (Integer 1)
      - `Task` (Complex 1)
      - `ID` (Integer 1)
      - `Subject` (String 1)
      - `DueDate` (Date 0-1)
      - `CompletedDate` (Date 0-1)
      - `Status` (String)
        - NotStarted #enum
        - In Progress #enum
        - Completed #enum
      - `Description` (String 0-1)
  - **Security:** auth token
  - **Protocol:** http

**Client receives:**
- `updateTaskResult`
  - **Contents:** none
  - **or:** `authFault`
  - **or:** `updateTaskFault`
4.3.3 Get Tasks

This service returns a list of employee’s tasks. The service signature is as follows:

**Type:** synchronous

**Client sends:**
```
getTasks
```

**Contents:**
```
  TaskDetails
    Employee
    ID
    Type
    Status
    TaskDetails
      Employee
      ID
      Subject
      Type
      DueDate
      Status
      NotStarted
      In Progress
      Completed
```

**Security:** auth token

**Protocol:** http

**Client receives:**
```
getTasksResult
```

**Contents:**
```
  TaskDetails
    Tasks
      Task
      ID
      Subject
      Type
      DueDate
      Status
      NotStarted
      In Progress
      Completed
```

In its simplest form, the client requests all tasks for a particular employee. The client can further specify which tasks are returned by adding additional query parameters. Type returns all tasks with a specified type. Status returns all tasks with the specified status or statuses.

4.3.4 Delete Task

This service deletes a task from an employee's task list. The service signature is as follows:

**Type:** synchronous

**Client sends:**
```
deleteTask
```

**Contents:**
```
  TaskDetails
    Employee
    ID
    Task
    ID
```

**Security:** auth token

**Protocol:** http

**Client receives:**
```
deleteTaskResult
```

**Contents:**
```
none
```

or:
```
authFault
```

or:
```
deleteTaskFault
```
4.3.5 Get Task

This service returns the details of a particular task. The service signature is as follows:

**Type:** synchronous

**Client sends:**
- `getTask`

  **Contents:**
  - TaskDetails (Complex 1)
    - Employee (Complex 1)
      - ID (Integer 1)
    - Task (Complex 1)
      - ID (Integer 1)
  - Security: auth token
  - Protocol: http

**Client receives:**
- `getTaskResult`

  **Contents:**
  - TaskDetails (Complex 1)
    - Task (String 1)
      - ID (Integer 1)
      - Subject (String 1)
      - Type (String 1)
      - DueDate (Date 0-1)
      - CompletedDate (Date 0-1)
      - Status (String)
        - NotStarted
        - In Progress
        - Completed
      - Description (String 0-1)

4.3.6 Portlets

When an employee is logged into the portal they will see the following task list portlet:

![Task List Portlet](image)

**Figure 37 - Task List Portlet**

Current tasks should show data returned from the getTasks service for not started and in progress status. Completed tasks should show data returned from this service for the completed status. Clicking on an Expense task should drive the Expense Report portlet to view the selected report. Clicking on a Review task should drive the Employee Review portlet to view the selected Review. Clicking on other tasks drives the Task List to show the Add/Update task portlet for that task. New Task shows the Add/Update portlet. Delete invokes deleteTask.
The add/update task portlet enables an employee to create or make modifications to a task:

![Add / Update Task Portlet](image)

Figure 38 - Add/Update Task Portlet

When the employee clicks on a task link in the task list, it should invoke getTask with the appropriate task id. The add / update task portlet should then be populated with the returned values. If activated through New Task all fields should be blank. On submission, if it is a new task the addTask operation should be invoked, otherwise updateTask should be used.

### 4.4 Browse Organization

The browse organization capability allows logged in employees to perform two actions:

1. Search for an employee by name or ID
2. Browse the departments of the organization

This involves invoking services provided by the HR System (the database schema and Web interface for management of this data is described in section 5.1). The interaction of the browse organization portlet, result views and the HR services are shown in the following diagram:

![HR Services](image)

Figure 39 - HR Services
The user may search for employees based on name or employee id by using the Browse Organization portlet Find Employee mechanism. This invokes the findEmployee service. This service can return no results, a single result or multiple results. In the event of a single result, the Employee View is shown, whereas multiple results leads to showing the Employee List. If there are no results the Browse Organization portlet remains visible. The employee view shows the details of a particular employee, their manager and any reportees.

The user may also browse departments by clicking on Department Listing. This invokes the getDepartments service to return a top level departments listing. Clicking on a department shows the Department View which is populated through a call to getDepartment. The result of this includes any child departments and employees of a department.

4.4.1 Find Employee

This service queries and returns employee information. The service signature is as follows:

Type: synchronous

Client sends: findEmployee

Contents: FindDetails
NameOrID String 1
Security: none
Protocol: http

Client receives: findEmployeeResult

Contents: FindResultDetails
Employee
ID Complex 1
Department Complex 1
ID Integer 1
Name String 1
CostCenter String 1
StreetAddress1 String 1
StreetAddress2 String 0-1
City String 1
State String 1
Zip String 1
Title String 1
Manager Complex 1
ID Integer 1
Name String 1
FirstName String 1
LastName String 1
Email String 1
BusinessPhone String 1
Reportees Complex 1
Employee Complex 0-*
ID Integer 1
FirstName String 1
LastName String 1

or Employees Complex 0-1
Employee Complex 0-*
ID Integer 1
FirstName String 1
LastName String 1
Email String 1
4.4.2 Get Employee

This service returns the details of a particular employee. The service signature is as follows:

**Type:** synchronous

**Client sends:**
- **getEmployee**
- **Contents:**
  - Employee
  - ID
- **Security:** none
- **Protocol:** http

**Client receives:**
- **getEmployeeResult**
- **Contents:**
  - EmployeeDetails
  - ID
  - Department
    - ID
    - Name
    - CostCenter
    - StreetAddress1
    - StreetAddress2
    - City
    - State
    - Zip
    - Title
    - Manager
      - ID
      - Name
      - FirstName
      - LastName
      - Email
      - BusinessPhone
    - Reportees
      - ID
      - FirstName
      - LastName

4.4.3 Get Departments

This service returns a list of departments based (optionally) on a parent department id. If no parent is given it returns all departments. The service signature is as follows:

**Type:** synchronous

**Client sends:**
- **getDepartments**
- **Contents:**
  - DepartmentDetails
    - ParentID
- **Security:** none
- **Protocol:** http
4.4.4 Get Department

This service returns a single department based on id. This also contains sub departments and list of employees in that department. The service signature is as follows:

**Type:** synchronous

**Client sends:**
`getDepartments`

*Contents:*
- `DepartmentDetails` Complex 1
  - ID Integer 0-1

*Security:* none

*Protocol:* http

**Client receives:**
`getDepartmentResult`

*Contents:*
- `DepartmentDetails` Complex 1
  - `Department` Complex 0-1
    - ID Integer 1
    - Name String 1
    - CostCenter String 1
    - StreetAddress1 String 1
    - StreetAddress2 String 0-1
    - City String 1
    - State String 1
    - Zip String 1
  - Contact Complex 0-1
    - ID Integer 1
    - FirstName String 1
    - LastName String 1
    - BusinessPhone String 1
    - Email String 1
  - Manager Complex 0-1
    - ID Integer 1
    - FirstName String 1
    - LastName String 1
    - BusinessPhone String 1
    - Email String 1
  - SubDepartments Complex 1
    - `Department` Complex 0-1
      - ID Integer 1
      - Name String 1
4.4.5 Portlets

The find employee and department listing functionality is accessed from the Browse Organization portlet:

![Browse Organization Portlet](image1)

Upon successfully looking up an employee or clicking on the Department Listing, the resultant portlet should be maximized. If the lookup fails (no employee or list returned) an appropriate message should be displayed. If employee lookup returns a list, the Employee List portlet should be shown:

![Employee List Portlet](image2)

If it returns a single employee, or the user drills down on a link in the list, the Employee View portlet is shown:

![Employee View Portlet](image3)
Similarly, the department listing link will change views to the Department List portlet, allowing drilldown through departments in the Department View portlet:

![Figure 43 - Department Listing Portlet](image)

![Figure 44 - Department View Portlet](image)

### 4.5 Manage Expense Reports

Expense reporting is a major function of all organizations. The Expense Report Process is a long lived workflow service that involves the employee, their manager and an accountant. This specification assumes that expense reporting has been carried out within the enterprise before through a basic web interface and the data structures for expense data already exist.

The goal of this use case is to create the back-end processing for expense reporting, utilizing new and existing services as appropriate, and create the front-end portal constructs that present these services to the user.

To understand the process, it is necessary to first describe the structure of an expense report, determine what states that report can be in and give an overview of the service interaction. Each operation is then further decomposed to the level of component or external services that cannot be further broken down.
4.5.1 Use Case Sequence

The actors, systems and services of the expense reporting use-case interact as shown in the following sequence diagram:

Figure 45 - Expense Reporting Sequence
4.5.2 Data Structures

The existing expense reporting database has the following structure:

![Expense Reporting Database Diagram]

Entities not shown are:

- Employee – from which EmployeeID, OverridingManagerID and AssignedAccountantID are linked in ExpenseReport
- Customer – from which CustomerID is linked in ExpenseReport

For implementation purposes the expense reporting database is part of the HR database that also contains Employees. This is documented in Appendix A.
4.5.3 State Diagram

Throughout its life, an expense report moves through a number of states. All of the actions (except for processing the payment) are invoked through the Employee Portal. The payment processing occurs as part of the backend processing for the whole Expense Report Process once the state of Accounting Approved is reached. In summary, the states are:

1. Created – Report exists within portal but not yet persisted or in process
2. Saved – Report has been persisted but is not yet in process
3. Submitted – Report has been submitted to processing awaits managerial approval
4. Manager Approved – Report is in processing and awaits accounting approval
5. Accounting Approved – Report is in processing and awaits payroll processing
6. Declined – Report has been actively declined by manager or accounting
7. Paid – Report has been processed by payroll, moves out of process and into archive

The transition between these states is managed through a long running process that handles the expense report from submission for processing until it is paid, and are activated through certain service operations.

Other operations, namely: getting a list of reports; getting a single report; and obtaining transactions from payroll; while part of the expense reporting service have no impact on state.

The state transitions can be summarized by the following diagram:

![Expense Reporting State Transitions Diagram]

Figure 47 - Expense Reporting State Transitions

4.5.4 Expense Reporting Services

The Expense Reporting Process is comprised of a number of actors, systems and services. All actors activate services through the employee portal. The different actors are:

1. Employee
2. Manager (also an employee)
3. Accountant (also an employee)
The employee portal interacts with two systems that provide services. The systems and services provided are:

4. HR System
   a. Get Reports
      Returns all reports for by employee, manager or accountant
   b. Get Report
      Returns details of a single report
   c. Get Item Types
      Returns list of valid types for report line items
   d. Save Report
      Persists report in Expense Report Database and sets status to Saved
   e. Submit Report
      Persists report in Expense Report Database and sets status to Submitted
   f. Authorize Report
      Sets status to either approved or declined for either manager or accountant
   g. Check Cut
      Activated by payroll once a check has been released

5. Payroll System
   a. Get Transactions
      Returns a list of unreported credit card transactions for an employee
   b. Cut Check
      Activates a physical process that releases a check to the employee

The interaction between the actors and systems through the portal and services can be summarized in the following diagram:
The following subsections define each of these services in sufficient detail that they may be implemented without requiring any specific implementation.

### 4.5.4.1 Get Reports

This service returns a list of expense reports for a particular employee. The service signature is as follows:

**Type:** synchronous

**Client sends:**

- *getReports*
  - *Contents:*
    - *GetReports*
      - employeeID: Integer 0-1
      - or managerID: Integer 0-1
      - or accountantID: Integer 0-1
  - Security: auth token
  - Protocol: http

**Client receives:**

- *listOfReports*
  - *Contents:*
    - *Reports*
      - *Report*
        - ID: Complex 1
        - Title: String 1
        - SubmittedDate: Date 1
        - Status: String 1
    - or authFault

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>getReports service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services, including remote security service, in sequence with control flow</td>
</tr>
<tr>
<td>Exception Handling and Compensating Services</td>
<td>handles authentication failed exception with a fault response</td>
</tr>
</tbody>
</table>

The process executed for this service is shown in the following diagram:
Figure 49 - Get Reports Service Process
4.5.4.2 Get Report

This service returns a single expense report instance given a specified report id. The service signature is as follows:

**Type:** synchronous

**Client sends:**
- **getReport**
  - **Contents:**
    - Report Complex 1
      - ID Integer 1
    - Security: auth token
    - Protocol: http

**Client receives:**
- **singleReport**
  - **Contents:**
    - Report Complex 1
      - ID Integer 1
      - Title String 1
      - Employee Complex 1
        - ID Integer 1
        - Name String 1
      - Status Complex 1
        - ID Integer 1
        - StatusText String 1
      - Dates Complex 1
        - Submitted Date 1
        - ManagerApproved Date 0-1
        - AccountingApproved Date 0-1
        - Paid Date 0-1
      - Payment Complex 0-1
        - ReferenceNumber String 1
        - Amount Decimal 1
      - Items Complex 1
        - Item Complex 0-*
          - ID Integer 1
          - Type Complex 1
            - ID Integer 1
            - TypeText String 1
          - Description String 1
          - Dates Complex 1
            - From Date 1
            - To Date 0-1
            - Days Integer 1
          - DailyRate Decimal 0-1
          - Total Decimal 1
          - Justification String 1
          - Adjustment Decimal 0-1
          - AdjustmentReason String 0-1
          - Receipts Complex 1
            - Required Boolean 1
            - Available Boolean 1
            - Verified Boolean 0-1
          - Mileage Complex 0-1
            - Miles Integer 1
            - Rate Decimal 1
            - Prepaid Decimal 0-1
or: authFault

The process executed for this service is shown in the following diagram:

![Diagram](image)

**Figure 50 - Get Report Service Process**

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Patterns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Services</td>
<td>getReport service is synchronous</td>
</tr>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services, including remote security service, in sequence with control flow</td>
</tr>
<tr>
<td>Exception Handling and</td>
<td>handles authentication failed exception with a fault response</td>
</tr>
<tr>
<td>Compensating Services</td>
<td></td>
</tr>
</tbody>
</table>
4.5.4.3 Get Item Types

This service returns a list of types for line items of expense reports. The service signature is as follows:

**Type:** synchronous

**Client sends:**
- **Contents:** get.ItemTypes
- **Security:** none
- **Protocol:** http

**Client receives:**
- **Contents:** list.ItemTypes
- **Type:** Complex 1
  - ID: Integer 1
  - Name: String 1

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>get.ItemTypes service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is shown in the following diagram:

![Figure 51 - Get Item Types Service Process](image-url)
4.5.4.4 Save Report

This service requests that a single expense report instance be stored without changing its status. The service signature is as follows:

**Type:** synchronous

**Client sends:**

<table>
<thead>
<tr>
<th>Contents:</th>
<th>saveReport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>Complex 1</td>
</tr>
<tr>
<td>ID</td>
<td>Integer 0-1</td>
</tr>
<tr>
<td>Title</td>
<td>String 1</td>
</tr>
<tr>
<td>Employee</td>
<td>Complex 1</td>
</tr>
<tr>
<td>ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Dates</td>
<td>Complex 1</td>
</tr>
<tr>
<td>Submitted</td>
<td>Date 1</td>
</tr>
<tr>
<td>Items</td>
<td>Complex 1</td>
</tr>
<tr>
<td>Item</td>
<td>Complex 0-*</td>
</tr>
<tr>
<td>ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Type</td>
<td>Complex 1</td>
</tr>
<tr>
<td>ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Description</td>
<td>String 1</td>
</tr>
<tr>
<td>Dates</td>
<td>Complex 1</td>
</tr>
<tr>
<td>From</td>
<td>Date 1</td>
</tr>
<tr>
<td>To</td>
<td>Date 0-1</td>
</tr>
<tr>
<td>Days</td>
<td>Integer 1</td>
</tr>
<tr>
<td>DailyRate</td>
<td>Decimal 0-1</td>
</tr>
<tr>
<td>Total</td>
<td>Decimal 1</td>
</tr>
<tr>
<td>Justification</td>
<td>String 1</td>
</tr>
<tr>
<td>Receipts</td>
<td>Complex 1</td>
</tr>
<tr>
<td>Required</td>
<td>Boolean 1</td>
</tr>
<tr>
<td>Available</td>
<td>Boolean 1</td>
</tr>
<tr>
<td>Mileage</td>
<td>Complex 0-1</td>
</tr>
<tr>
<td>Miles</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Rate</td>
<td>Decimal 1</td>
</tr>
</tbody>
</table>

**Security:** auth token

**Protocol:** http

**Client receives:**

<table>
<thead>
<tr>
<th>Contents:</th>
<th>saveConfirmation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>Complex 1</td>
</tr>
<tr>
<td>ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Title</td>
<td>String 1</td>
</tr>
</tbody>
</table>

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>getReport service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database and perform processing</td>
</tr>
</tbody>
</table>
Serial Service Orchestration executes services, including remote security service, in sequence with control flow.

Exception Handling and Compensating Services handles authentication failed exception and database update exception with a fault response.

The process executed for this service is shown in the following diagram:

Figure 52 - Save Report Service Process
4.5.4.5 Submit Report

This service requests that a single expense report instance be stored AND that its status is changed to Submitted. If this service is successful, the report will appear on the employee’s manager’s task list. The service signature is as follows:

**Type:** synchronous

**Client sends:** submitReport

*Contents:*

<table>
<thead>
<tr>
<th>ID</th>
<th>Complex 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Integer  0-1</td>
</tr>
<tr>
<td>Employee ID</td>
<td>String 1</td>
</tr>
<tr>
<td>Dates</td>
<td>Complex 1</td>
</tr>
<tr>
<td>Submitted</td>
<td>Date 1</td>
</tr>
<tr>
<td>Items</td>
<td>Complex 1</td>
</tr>
<tr>
<td>Item ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Type</td>
<td>Complex 1</td>
</tr>
<tr>
<td>Description</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Dates From</td>
<td>String 1</td>
</tr>
<tr>
<td>To</td>
<td>Date 1</td>
</tr>
<tr>
<td>Days</td>
<td>Integer 1</td>
</tr>
<tr>
<td><strong>DailyRate</strong></td>
<td>Decimal 0-1</td>
</tr>
<tr>
<td>Total</td>
<td>Decimal 1</td>
</tr>
<tr>
<td>Justification</td>
<td>String 1</td>
</tr>
<tr>
<td>Receipts</td>
<td>Complex 1</td>
</tr>
<tr>
<td>Required</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Available</td>
<td>String 1</td>
</tr>
<tr>
<td><strong>Mileage</strong></td>
<td>Complex 0-1</td>
</tr>
<tr>
<td>Miles</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Rate</td>
<td>Decimal 1</td>
</tr>
<tr>
<td>Prepaid</td>
<td>Decimal 0-1</td>
</tr>
</tbody>
</table>

**Security:** auth token

**Protocol:** http

**Client receives:** submitConfirmation

*Contents:*

<table>
<thead>
<tr>
<th>ID</th>
<th>Complex 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Integer 1</td>
</tr>
</tbody>
</table>

or:

**authFault**

or:

**submitFailedFault**

*Contents:* Description of failure reason

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Synchronous Services &amp; Asynchronous Services</th>
<th>submitReport service is mostly synchronous but after returning a response, asynchronously sends the email and assigns the task</th>
</tr>
</thead>
</table>
Component uses component services to access database, perform processing, send email and create and assign a task to the manager

Serial Service Orchestration & Parallel Service Orchestration executes services, including remote security service, in sequence with control flow and in parallel for email and task management

Exception Handling and Compensating Services handles authentication failed exception and database update exception with a fault response

The process executed for this service is shown in the following diagram:

![Diagram](image-url)

Figure 53 - Submit Report Service Process
4.5.4.6 Authorize Report

This service requests that a single expense report be authorized. It can be authorized by either a managerID or accountantID. The service signature is as follows:

**Type:** synchronous

**Client sends:**

- **authorizeReport**
  - **Contents:**
    - ReportAuthorization Complex 1
      - ID Integer 1
      - Title String 1
      - Employee Complex 1
        - ID Integer 1
      - Authorization Complex 1
        - ManagerID Integer 0-1
        - or AccountantID Integer 0-1
      - Status String 1
        - Approved #enum
        - or Declined #enum
      - Declination Reason String 0-1
      - Items Complex 0-1
        - Item Complex 0-*
          - ID Integer 1
      - Receipts Complex 1
        - Verified Boolean 1

**Security:** auth token

**Protocol:** http

**Client receives:**

- **authorizationConfirmation**
  - **Contents:** Report Complex 1
    - ID Integer 1
    - Title String 1

  or: **authFault**

  or: **authorizationFailedFault**
  - **Contents:** Description of failure reason

**Patterns Covered:**

| Synchronous Services & Asynchronous Services | authorizeReport service is mostly synchronous but after returning a response, asynchronously sends the email and assigns the task, or invokes the asynchronous cutCheck service |
| Component | uses component services to access database, perform processing, send email and create and assign a tasks |
| Serial Service Orchestration & Parallel Service Orchestration | executes services, including remote security service and cut check, in sequence with control flow |
| Exception Handling and Compensating Services | handles authentication failed exception and database exception with a fault response |
The process executed for this service is shown in the following diagram:

![Diagram showing the authorize report service process](image)

Figure 54 - Authorize Report Service Process
4.5.4.7 Check Cut

This service is invoked as a callback from the Payroll system when a check has been processed. The service signature is as follows:

**Type:** asynchronous

**Client sends:**
- **Contents:**
  - PaymentNotification Complex 1
  - Report Complex 1
  - ID Integer 1
  - Payment Complex 1
  - ReferenceNumber String 1
  - Amount Decimal 1

**Security:** http

**Protocol:** http

**Client receives:** none

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>checkCut service is asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to perform processing, access database, remove task and send email</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is shown in the following diagram:

![Check Cut Service Process Diagram](image-url)

Figure 55 – Check Cut Service Process
4.5.4.8 Get Transactions

This payroll service is invoked from the portal to obtain a list of credit card transactions for a particular employee. The service signature is as follows:

**Type:** synchronous

**Client sends:**

- `getTransactions`
  - **Contents:**
    - `TransactionDetails` Complex 1
    - `Employee` Complex 1
    - `ID` Integer 1
    - `Dates` Complex 1
    - `From` Date 1
    - `To` Date 1
  - **Security:** https + authCode + signing
  - **Protocol:** https

**Client receives:**

- `listOfTransactions`
  - **Contents:**
    - `Transactions` Complex 1
      - `Transaction` Complex 0- *
      - `Date` Date 1
      - `Description` String 1
      - `Amount` Decimal 1
  - or: `authFault`
  - or: `signatureIncorrectFault`

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Services</td>
<td>getTransactions service is synchronous</td>
</tr>
<tr>
<td>Component</td>
<td>uses component to retrieve information from the Payroll mainframe system</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

The process executed for this service is shown in the following diagram:
Figure 56 – Get Transactions Service Process
4.5.4.9 Cut Check

This payroll service is invoked from the authorization service start the process of cutting a check for the expenses amount. On completion this service will invoke the Check Cut service. The service signature is as follows:

**Type:** asynchronous

**Client sends:** cutCheck  
*Contents:* CheckDetails  
- Complex 1  
- Payee  
- Complex 1  
- ID  
- Integer 1  
- Full Name  
- String 1  
- Address  
- Complex 1  
- Street  
- String 1  
- City  
- String 1  
- State  
- String 1  
- Zip  
- String 1  
- Amount  
- Decimal 1  
- CheckCutServiceAccessPoint  
- String 1  

**Security:** https + authCode + signing  
**Protocol:** https

**Client receives:** none

The process executed for this service is shown in the following diagram:

![Diagram of Cut Check Service Process]

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>cutCheck service is asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component to invoke and wait for processing in the Payroll mainframe system</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence</td>
</tr>
</tbody>
</table>
4.5.4.10 Overall Workflow Service

The services that comprise the HR system are partly stateless and partly stateful as shown in this diagram:

![Diagram showing HR System services]

**Figure 58 – Statefulness of Expense Reporting Services**

The services to get a list of reports, a single report and item types and to save the report do not play any part in the overall workflow.

The submit report service initiates the workflow. Report authorization, whether it is approved or declined, from manager or accountant, continues the workflow. The workflow finally finishes when the check is cut by the payroll system and the report transitions to the Paid state.

4.5.5 Portlet Implementation

The portion of the employee portal dealing with expense reporting is required to interact with the Expense Reporting system solely through its exposed services. The portlets involved are required to transition through a page flow to guide the user in the process.

The report should be built up in session state within the portal. It is only persisted when the user activates the Save Report or Submit Report services. Similar when a list of reports or single report is retrieved using the Get Reports or Get Report services the report data should be stored locally within the user’s session.
Expense reporting actions can be initiated through the Expense Report portlet or through the employee’s task list. The task list will activate the Get Report service to show the View Report or Authorize Report portlets as appropriate.

The Authorize Report portlet has two different flavors, depending on whether the user is the employee’s manager or an accountant.

The overall portlet flow is shown in the following diagram:

![Expense Reporting Portlet Flow Diagram](image)

Figure 59 - Expense Reporting Portlet Flow

The initial normal portlet size (allowing other portlets to be visible) should be used for viewing the list of reports returned by Get Reports in either the My Reports (by employee) or Staff Reports (by manager / accountant).

When operations are carried out on a single report the portlet should be maximized to give the optimal working space.
4.5.5.1 View Report

The View Report portlet should be maximized and have the following construction:

![View Report Diagram]

Only visible if Report is in New, Saved or Declined status.

4.5.5.2 Update Report

The Update Report portlet should be maximized and have the following construction:

![Update Report Diagram]
4.5.5.3 Create Report

The Create Report portlet should be maximized and have the following construction:

![Create Report Portlet](image)

4.5.5.4 Choose Transactions

The Choose Transactions portlet should be maximized and have the following construction:

![Choose Transactions Portlet](image)
4.5.5.5 Add /Edit Line Item

The Add/Edit Line Item portlet should be maximized and have the following construction:

![Add / Edit Line Item](image1)

4.5.5.6 Authorize Report (Managers View)

This Authorize Report portlet should be maximized and have the following construction:

![Authorize Report](image2)
4.5.5.7 Authorize Report (Accountants View)

This Authorize Report portlet should be maximized and have the following construction:

<table>
<thead>
<tr>
<th>Name</th>
<th>Stewart McLeod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>GenCo Conference</td>
</tr>
<tr>
<td>Purpose</td>
<td>Attend conference</td>
</tr>
<tr>
<td>Submitted Date</td>
<td>3/5/2004</td>
</tr>
<tr>
<td>Manager Approved</td>
<td>3/12/2004</td>
</tr>
<tr>
<td>Manager Approved</td>
<td>Not Yet</td>
</tr>
<tr>
<td>Total</td>
<td>$267.63</td>
</tr>
<tr>
<td>Status</td>
<td>Manager Approved</td>
</tr>
</tbody>
</table>

### Line Items

<table>
<thead>
<tr>
<th>Type</th>
<th>From</th>
<th>To</th>
<th>Description</th>
<th>Days</th>
<th>Daily Rate</th>
<th>Total</th>
<th>Receipts R P V</th>
<th>Group</th>
<th>Adjusted Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air</td>
<td>3/2/2004</td>
<td>3/5/2004</td>
<td>Team XXX in SEA round-trip</td>
<td>3</td>
<td>N/A</td>
<td>267.65</td>
<td>R P</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.5.5.8 Authorize Line Item

The Authorize Line Item portlet should be maximized and have the following construction:

<table>
<thead>
<tr>
<th>Name</th>
<th>Stewart McLeod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>GenCo Conference</td>
</tr>
<tr>
<td>Purpose</td>
<td>Attend conference</td>
</tr>
</tbody>
</table>

### Description

From SIC to SEA round-trip

### Justification

Cheapest air fare

### Type

Air Fare

### From

3/2/2004

### To

3/5/2004

### Days

3

### Rate

N/A

### Total

$267.65

### Group

1

### Receipt Required

✓

### Have Receipt

✓

### Prepaid by Company

✓

### Mileage Rate

N/A

### Prepaid Amount

$267.65

### Receipt Verified

✓

### Adjustment

$10

### Reason

Receipt only for $257.05

### Approve

✓

### Approve and Next

✓

### Decline

✓

Receipt verification is not enabled if no receipt is required.
4.5.5.9 Portlet and Service Interaction

The usage of services by portlets and the actions within the portlet flow that controls them can be summarized by the following diagram:

![Portlet and Service Interaction Diagram](image)

Figure 60 - Portlet and Service Interaction

4.6 Manage Reviews

Employee Review management is another major function of all organizations. The Employee Review Process is a long lived workflow service that involves the employee, their manager and an HR Representative. This assumes that automated employee review is a new process that is replacing a previous paper based process.

The goal of this use-case is to create the back-end processing for employee reviews, define new services, and create the front-end portal constructs that present these services to the user.

To understand the process, it is necessary to first describe the sequence of events in employee reviews, specify the required data structures, determine what states the review can be in and give an overview of the service interaction. Each operation is then further decomposed to the level of component or external services that cannot be further broken down.
4.6.1 Use Case Sequence

The actors, systems and services of the employee review use-case interact in two distinct phases. The first phase is the initial creation of the review criteria, which occurs at the beginning of the quarter. This is shown in the following diagram:

Figure 61 - Employee Review Sequence Phase I

During this phase the manager creates initial review and adds criteria against which the employee will be judged during that quarter. The employee and HR rep may both accept and agree to these criteria. Some of these criteria may be related to a salary adjustment, and some to a bonus allocation. The performance against these criteria is rated at the end of the quarter and the bonus / salary change applied accordingly.
At quarter end the next phase is initiated by the timer event. The interaction of actors and systems in this phase is shown in the following diagram:

During this phase the Manager receives the initial notification that the review needs to be completed. They will retrieve the report and add in employee ratings and comments for each criterion previously determined. The employee will then be notified and verify that they agree with the each of the ratings. They can dispute and cycle back to the manager or agree at which point the HR Rep needs to determine if salary changes or bonus payments are required.

Changes to salary and/or bonus payments are achieved through services provided in the Payroll system. Once all of this is complete the review cycle is finished and the employee is notified of any salary changes or bonus payments.
4.6.2 Data Structures

The employee review database has the following structure:

![Database Structure Diagram]

The Review table is linked to the Employee table in the HR database through the employee id.

4.6.3 State Diagram

Throughout its life, an expense report moves through a number of states. All of the actions (except for processing the payment) are invoked through the employee portal. The payment processing occurs as part of the backend processing for the whole Expense Report Process once the state of Accounting Approved is reached. In summary the states are:

1. Created – Review exists within portal but not yet persisted or in process
2. Saved – Review has been persisted but is not yet in process
3. ConfirmCriteria – Manager has created review criteria, but not yet agreed by employee
4. CriteriaDisputed – Employee does not accept criteria
5. CriteriaConfirmed – Employee accepts criteria
6. CriteriaAgreed – HR Rep agrees with criteria
7. Rated – At Quarter end Manager rates Employee based on Criteria
8. RatingsDisputed – Employee does not accept ratings
9. RatingsConfirmed – Employee accepts ratings
10. Processing – HR Rep defines compensation changes and submits for processing
11. Completed – Any compensation changes have been made, end of review cycle

The transition between these states is managed through a long running process that handles the employee review from submission for processing until it is completed, and are activated through certain service operations.
The state transitions can be summarized by the following diagram:

Figure 64 - Employee Review State Transitions

4.6.4 Employee Review Services

The Employee Review Process is comprised of a number of actors, systems and services. All actors activate services through the employee portal. The different actors are:

1. Employee
2. Manager (also an employee)
3. HR Representative (also an employee)

The employee portal interacts with two systems that provide services. The systems and services provided are:

1. Review System
   a. Get Reviews
      Returns all reviews by employee, manager or hr rep
   b. Get Review
      Returns details of a single review
c. Save Review
   Persists review in Employee Review Database and sets status to Saved

d. Submit Review
   Persists review in Employee Review Database and sets status to Confirm Criteria

e. Verify Criteria
   Sets status to Criteria Confirmed, Agreed or Disputed

f. Rate Review
   Adds rating information to review and sets status to Rated

g. Verify Rating
   Sets status to Rating Confirmed or Disputed

h. Process Review
   Requests payroll to make compensation changes recommended by hr rep

i. Salary Updated
   Callback service invoked when salary updates are completed

j. Bonus Paid
   Callback service invoked when bonus payment is complete

2. Payroll System
   a. Update Salary
      Activates a process to updates an employees salary
   b. Pay Bonus
      Activates a physical process to create a bonus payment check

The interaction between the actors and systems through the portal and services can be summarized in the following diagram:

![Diagram of Employee Review Services]

Figure 65 - Employee Review Services

The following subsections define each of these services in sufficient detail that they may be implemented without requiring any specific implementation.
4.6.4.1 Get Reviews

This service returns a list of employee reviews for a particular employee, their manager or overriding manager, or hr representative. The service signature is as follows:

Type: synchronous

Client sends:

- Contents: getReviews
  - GetReviews
  - employeeID or managerID or hrRepId

Security: auth token
Protocol: http

Client receives:

- Contents: listOfReviews
  - Reviews
    - ID
    - Employee ID
    - FirstName
    - LastName
    - BonusReview
    - StartDate

or:

authFault

Patterns Covered:

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>getReviews service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services, including remote security service, in sequence with control flow</td>
</tr>
<tr>
<td>Exception Handling and Compensating Services</td>
<td>handles authentication failed exception with a fault response</td>
</tr>
</tbody>
</table>

The process executed for this service is shown in the following diagram:
Figure 66 - Get Reviews Service Process
4.6.4.2 Get Review

This service returns a single employee review instance given a specified review id. The service signature is as follows:

**Type:** synchronous

**Client sends:**

<table>
<thead>
<tr>
<th>Contents:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Security:</td>
<td>auth token</td>
</tr>
<tr>
<td>Protocol:</td>
<td>http</td>
</tr>
</tbody>
</table>

**Client receives:**

<table>
<thead>
<tr>
<th>Contents:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Employee ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>FirstName</td>
<td>String 1</td>
</tr>
<tr>
<td>LastName</td>
<td>String 1</td>
</tr>
<tr>
<td>Status ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Status Text</td>
<td>String 1</td>
</tr>
<tr>
<td>Dates StartDate</td>
<td>Date 1</td>
</tr>
<tr>
<td>CompletedDate</td>
<td>Date 0-1</td>
</tr>
<tr>
<td>BonusReview</td>
<td>Boolean 1</td>
</tr>
<tr>
<td>Comments</td>
<td>Complex 1</td>
</tr>
<tr>
<td>ManagerComments</td>
<td>String 0-1</td>
</tr>
<tr>
<td>EmployeeComments</td>
<td>String 0-1</td>
</tr>
<tr>
<td>OverallRating</td>
<td>Decimal 0-1</td>
</tr>
<tr>
<td>Items Item</td>
<td>Complex 0-*</td>
</tr>
<tr>
<td>ID</td>
<td>Integer 1</td>
</tr>
<tr>
<td>Description</td>
<td>String 1</td>
</tr>
<tr>
<td>Weighting</td>
<td>Integer 1</td>
</tr>
<tr>
<td>BonusItem</td>
<td>Boolean 1</td>
</tr>
<tr>
<td>Rating</td>
<td>Integer 0-1</td>
</tr>
<tr>
<td>Goals</td>
<td>String 0-1</td>
</tr>
<tr>
<td>Comments</td>
<td>Complex 1</td>
</tr>
</tbody>
</table>

or:

| auth Fault |

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>getReview service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services, including remote security service, in sequence with control flow</td>
</tr>
<tr>
<td>Exception Handling and Compensating Services</td>
<td>handles authentication failed exception with a fault response</td>
</tr>
</tbody>
</table>
The process executed for this service is shown in the following diagram:

Figure 67 - Get Review Service Process
### 4.6.4.3 Save Review

This service requests that a single review instance be stored without changing its status. The service signature is as follows:

**Type:** synchronous

**Client sends:**

- *Contents:* `saveReview`
  - ID
  - Employee
    - ID
    - FirstName
    - LastName
  - Manager
    - ID
  - Dates
    - StartDate
  - BonusReview
    - Items
      - Item
        - ID
        - Description
        - Weighting
        - BonusItem
      - Goals
  - Security: auth token
  - Protocol: http

**Client receives:**

- *Contents:* `saveConfirmation`
  - ID

or:

- *Contents:* `authFault`

or:

- *Contents:* `saveFailedFault`
  - Description of failure reason

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Synchronous Services</th>
<th>saveReview service is synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to access database and perform processing</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services, including remote security service, in sequence with control flow</td>
</tr>
<tr>
<td>Exception Handling and Compensating Services</td>
<td>handles authentication failed exception and database update exception with a fault response</td>
</tr>
</tbody>
</table>

The process executed for this service is shown in the following diagram:
Figure 68 - Save Review Service Process
4.6.4.4 Submit Review

This service requests that an employee review instance be stored AND that its status is changed to confirm criteria. If this service is successful, the review will appear on the employee’s task list. The service signature is as follows:

**Type:** synchronous

**Client sends:** submitReview

**Contents:**
- Review
  - ID
  - Employee
    - ID
    - First Name
    - Last Name
  - Manager
    - ID
  - Dates
    - Start Date
    - Bonus Review
  - Items
    - Item
      - ID
      - Description
      - Weighting
      - Bonus Item
  - Goals

**Security:** auth token

**Protocol:** http

**Client receives:** submitConfirmation

**Contents:**
- Review
  - ID

or:

**Contents:** authFault

or:

**Contents:** submitFailedFault

Patterns Covered:

| Synchronous Services & Asynchronous Services | submitReview service is mostly synchronous but after returning a response, asynchronously sends the email and assigns the task |
| Component | uses component services to access database, perform processing, send email and create and assign a task to the manager |
| Serial Service Orchestration & Parallel Service Orchestration | executes services, including remote security service, in sequence with control flow and in parallel for email and task management |
| Exception Handling and Compensating Services | handles authentication failed exception and database exception with a fault response |
The process executed for this service is shown in the following diagram:

![Diagram of Submit Review Service Process](image)

**Figure 69 - Submit Review Service Process**

### 4.6.4.5 Verify Criteria

This service verifies the criteria of an employee review instance. The service may be invoked by an employee or hr representative. If this service is successful, the review status will change to criteria confirmed or disputed, and it will appear on the hr reps task list or move into the end of quarter wait stage. The service signature is as follows:
Type: synchronous

**Client sends:** verifyCriteria

Contents:
- Review
  - ID
  - Employee
    - ID
  - or HRRep
    - ID
- CriteriaReview
  - Accepted #enum
  - or Disputed #enum
- DisputeReason

Security: auth token
Protocol: http

**Client receives:** verifyConfirmation

Contents:
- Review
  - ID

**or:** authFault

**or:** verifyFailedFault

Contents: Description of failure reason

Patterns Covered:

| Synchronous Services & Asynchronous Services | verifyCriteria service is mostly synchronous but after returning a response, asynchronously sends the email and assigns the task |
| Component | uses component services to access database, perform processing, send email and create and assign a task to the manager |
| Serial Service Orchestration | executes services, including remote security service, in sequence with control flow and in parallel for email and task management |
| Exception Handling and Compensating Services | handles authentication failed exception and database update exception with a fault response |

The process executed for this service is shown in the following diagram:
Figure 70 - Verify Criteria Service Process

Figure 71 - Quarter End Timer Process
4.6.4.6 Rate Review

This service adds rating information to an employee review instance. If this service is successful, the review status will change to rated, and it will appear on the employee’s task list. The service signature is as follows:

**Type:** synchronous

**Client sends:** rateReview

**Contents:**
- Review
  - ID Integer 1
  - Employee Complex 1
  - Manager Complex 1
  - Comments String 1
  - Items Complex 1
    - Item
      - ID Integer 1
      - Rating Integer 1
    - ManagerComments String 0-1
- Security: auth token
- Protocol: http

**Client receives:** ratingConfirmation

**Contents:**
- Review Complex 1
  - ID Integer 1

or:

- authFault

or:

- ratingFailedFault

**Patterns Covered:**

| Synchronous Services and Asynchronous Services | rateReview service is mostly synchronous but after returning a response, asynchronously sends the email and assigns the task |
| Component | uses component services to access database, perform processing, send email and create and assign a task to the manager |
| Serial Service Orchestration | executes services, including remote security service, in sequence with control flow and in parallel for email and task management |
| Exception Handling and Compensating Services | handles authentication failed exception and database update exception with a fault response |

The process executed for this service is shown in the following diagram:
Figure 72 - Rate Review Service Process
4.6.4.7 Verify Rating

This service verifies the rating information of an employee review instance. The service may be invoked by an employee. If this service is successful, the review status will change to rating confirmed or disputed, and it will appear on the hr reps task list. The service signature is as follows:

**Type:** synchronous

**Client sends:** verifyRating

*Contents:* Review

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer</td>
<td>1</td>
</tr>
<tr>
<td>Employee</td>
<td>Complex</td>
<td>1</td>
</tr>
<tr>
<td>ID</td>
<td>Integer</td>
<td>1</td>
</tr>
<tr>
<td>Comments</td>
<td>String</td>
<td>0-1</td>
</tr>
<tr>
<td>RatingReview</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>Accepted</td>
<td>#enum</td>
<td></td>
</tr>
<tr>
<td>or Disputed</td>
<td>#enum</td>
<td></td>
</tr>
<tr>
<td>DisputeReason</td>
<td>String</td>
<td>0-1</td>
</tr>
<tr>
<td>Items</td>
<td>Complex</td>
<td>1</td>
</tr>
<tr>
<td>Item</td>
<td>Complex</td>
<td>0-*</td>
</tr>
<tr>
<td>ID</td>
<td>Integer</td>
<td>1</td>
</tr>
<tr>
<td>Rating</td>
<td>Integer</td>
<td>1</td>
</tr>
<tr>
<td>EmployeeComments</td>
<td>String</td>
<td>0-1</td>
</tr>
</tbody>
</table>

**Security:** auth token

**Protocol:** http

**Client receives:** verifyConfirmation

*Contents:* Review

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer</td>
<td>1</td>
</tr>
</tbody>
</table>

or:

**Contents:** authFault

or:

**Contents:** verifyFailedFault

*Contents:* Description of failure reason

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Pattern Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Services &amp; Asynchronous Services</td>
<td>verifyRating service is mostly synchronous but after returning a response, asynchronously sends the email and assigns the task</td>
</tr>
<tr>
<td>Component</td>
<td>uses component services to access database, perform processing, send email and create and assign a task</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services, including remote security service, in sequence with control flow and in parallel for email and task management</td>
</tr>
<tr>
<td>Exception Handling and Compensating Services</td>
<td>handles authentication failed exception and database update exception with a fault response</td>
</tr>
</tbody>
</table>
The process executed for this service is shown in the following diagram:
4.6.4.8 Process Review

This service starts the processing of any compensation change indicated by an employee review instance. The service may be invoked by an hr representative. If this service is successful, the review status will change to Processing, while it waits for confirmation of changes, followed by completed. The service signature is as follows:

**Type:** synchronous

**Client sends:**
- `processReview`

  **Contents:**
  - `Review`
    - `ID` Integer 1
    - `HRRep`
      - `ID` Integer 1
    - `SalaryChange`
      - `Decimal` 0-1
    - `BonusPayment`
      - `Decimal` 0-1

  **Security:** auth token
  **Protocol:** http

**Client receives:**
- `processConfirmation`
  **Contents:**
  - `Review`
    - `ID` Integer 1

or:
- `authFault`

or:
- `processReviewFault`
  **Contents:** Description of failure reason

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Services and Asynchronous Services</td>
<td>processReview service is mostly synchronous but after returning a response, asynchronously sends the email and assigns the task</td>
</tr>
<tr>
<td>Component</td>
<td>uses component services to access database, perform processing, send email and create and assign a task to the manager</td>
</tr>
<tr>
<td>Serial Service Orchestration &amp; Parallel Service Orchestration</td>
<td>executes services, including remote security service, in sequence with control flow and in parallel for email and task management</td>
</tr>
<tr>
<td>Exception Handling and Compensating Services</td>
<td>handles authentication failed exception and database update exception with a fault response</td>
</tr>
</tbody>
</table>

The process executed for this service is shown in the following diagram:
4.6.4.9 Update Salary

This service is invoked on the payroll system to initiate changing an employee's salary. Since the payroll system is an external system, the actual processing could be a manual or long term process, which is invoked asynchronously. The service signature is as follows:

**Type:** asynchronous

**Service sends:** updateSalary

**Contents:**
- UpdateRequest
  - UniqueID: Complex, 1
  - Employee: Complex, 1
  - ID: Integer, 1
  - SSN: String, 1
  - DateOfBirth: Date, 1
**Patterns Covered:**

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>updateSalary service is asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component to invoke and wait for processing in the Payroll mainframe system</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence</td>
</tr>
</tbody>
</table>

The process executed for this service is shown in the following diagram:

![Figure 75 - Update Salary Service Process](image)

**4.6.4.10 Salary Updated**

This service is invoked as a callback from the Payroll system when a salary update has been processed. The service signature is as follows:

**Type:** asynchronous

**Client sends:** salaryUpdated

**Contents:**
- UpdateNotification
- UniqueID
- EmployeeID
- SalaryChange
- NewSalary

**Security:** http
Protocol: http

Client receives: none

Patterns Covered:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous Services</td>
<td>salaryUpdated service is asynchronous</td>
</tr>
<tr>
<td>Component</td>
<td>uses component services to perform processing, access database, remove task and send email</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>

### 4.6.4.11 Pay Bonus

This service is invoked on the payroll system to initiate paying a bonus to an employee. Since the payroll system is an external system, the actual processing could be a manual or long term process, which is invoked asynchronously. The service signature is as follows:

Type: asynchronous

Client sends: payBonus

**Contents:**

- BonusRequest
  - UniqueID
  - Employee
    - ID
    - SSN
    - DateOfBirth
    - Address
      - Street
      - City
      - State
      - Zip
    - BonusAmount

Security: https + authCode + signing

Protocol: https

Client receives: none

or: authFault

or: updateFault

**Contents:** Description of failure reason

Patterns Covered:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous Services</td>
<td>payBonus service is asynchronous</td>
</tr>
<tr>
<td>Component</td>
<td>uses component to invoke and wait for processing in the Payroll mainframe system</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence</td>
</tr>
</tbody>
</table>
The process executed for this service is shown in the following diagram:

### 4.6.4.12 Bonus Paid

This service is invoked as a callback from the Payroll system when a bonus payment has been processed. The service signature is as follows:

**Type:** asynchronous

**Client sends:** `bonusPaid`

- **Contents:**
  - `BonusNotification`
  - `UniqueID` (Integer 1)
  - `EmployeeID` (Integer 1)
  - `BonusAmount` (Decimal 1)

**Security:** http  
**Protocol:** http

**Client receives:** none

**Patterns Covered:**

<table>
<thead>
<tr>
<th>Asynchronous Services</th>
<th>bonusPaid service is asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>uses component services to perform processing, access database, remove task and send email</td>
</tr>
<tr>
<td>Serial Service Orchestration</td>
<td>executes services in sequence with control flow</td>
</tr>
</tbody>
</table>
4.6.5 Portlet Implementation

The portion of the employee portal dealing with expense reporting is required to interact with the Expense Reporting system solely through its exposed services. The portlets involved are required to transition through a page flow to guide the user in the process.

The report should be built up in session state within the portal. It is only persisted when the user activates the Save Report or Submit Report services. Similarly, when a list of reports or single report is retrieved using the Get Reports or Get Report services the report data should be stored locally within the user's session.

Expense reporting actions can be initiated through the Expense Report portlet or through the employee's task list. The task list will activate the Get Report service show the View Report or Authorize Report portlets as appropriate.

The Authorize Report portlet has two different flavors, depending on whether the user is the employee's manger or an accountant.

The overall portlet flow is shown in the following diagram:

![Employee Review Portlet Flow Diagram](image)

Figure 76 – Employee Review Portlet Flow

The initial normal portlet size (allowing other portlets to be visible) should be used for viewing the list of reports returned by Get Reviews in either the My Reviews (by employee) or Staff Reviews (by manager/hr rep).

When operations are carried out on a single report the portlet should be maximized to give the optimal working space.
4.6.5.1 Create Review

![Create Review Diagram]

4.6.5.2 Update Review

![Update Review Diagram]
4.6.5.3 Verify Criteria

Employee Reviews

Verify Criteria

<table>
<thead>
<tr>
<th>Employee</th>
<th>Employee Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bonus Review?</td>
<td>✓</td>
</tr>
</tbody>
</table>

Accept | Dispute | Cancel

Dispute Reason

Criteria

<table>
<thead>
<tr>
<th>Enter Text</th>
<th>Enter More Text</th>
<th>Weight</th>
<th>For Bonus?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enter Text</th>
<th>Enter More Text</th>
<th>Weight</th>
<th>For Bonus?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enter Text</th>
<th>Enter More Text</th>
<th>Weight</th>
<th>For Bonus?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

4.6.5.4 Rate Review

Employee Reviews

Rate Review

<table>
<thead>
<tr>
<th>Employee</th>
<th>Employee Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bonus Review?</td>
<td>✓</td>
</tr>
<tr>
<td>Manager Override</td>
<td>(managers id)</td>
</tr>
</tbody>
</table>

Submit Review | Save Review

Overall comments

Criteria

<table>
<thead>
<tr>
<th>Enter Text</th>
<th>Enter More Text</th>
<th>Weight</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enter Text</th>
<th>Enter More Text</th>
<th>Weight</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enter Text</th>
<th>Enter More Text</th>
<th>Weight</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
4.6.5.5 Verify Rating

4.6.5.6 Change Compensation
5  SUPPORTING APPLICATIONS

5.1 HR Application

In order to support the security and expense reporting functions a simple HR application is required that allows management of Departments, Employees and Titles. This is a basic CRUD application with pages to facilitating viewing, adding, updating, and deleting each entity.

The employees and the definition of their managers are used within the expense reporting application. The add / update employee functionality and employee role management is required to integrate with the Security application as defined in 2.3 and 2.5.2.

The employee add / update / view pages should retrieve employee and possible systems through the getEmployeeSystems and getPossibleSystems services. When an employee is added or updated an employeeChange message should be published on the hrChange Topic.

The edit employee roles page should retrieve employee and possible roles for that employee through the getEmployeeRoles and getPossibleRoles services. When employee roles are changes a rolesChange message should be published on the hrChange Topic.

5.1.1 HR Application Data Structures

The HR application is supported by the following data structure in the HR database:

![HR Database Diagram](image-url)

---

Figure 77 - HR Database
5.1.2 HR Page Flow

The HR application should consist of a number of Web pages interconnected by a page flow defined as follows:

![HR Web Site Page Flow Diagram]

Figure 78 - HR Web Site Page Flow
5.1.3 Maintain Departments

5.1.4 Add Department
5.1.5 Edit Department

5.1.6 View Department
5.1.7 Maintain Employees

<table>
<thead>
<tr>
<th>Employee Name</th>
<th>Department</th>
<th>Manager</th>
<th>Date of Birth</th>
<th>Work Phone</th>
<th>Email</th>
<th>View</th>
<th>Edit</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>(firstname + lastname)</td>
<td>(see notes)</td>
<td>(see notes)</td>
<td>(dateofbirth)</td>
<td>(businessphone)</td>
<td>(email)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Records x-y of nnn (Actual mmm)
5.1.8 Add Employee

5.1.9 Edit Employee
5.1.10 Employee Roles

5.1.11 View Employee
5.1.12 Maintain Titles

Maintain Titles

Previous button takes the user back one page. Only visible if this is not the first page.

Details of how which records are showing and how many records there are. If there are more than 500 records, nnn will be 500 and the actual number mmm will be shown.

Next button takes the user forward one page. Only visible if this is not the last page.

Takes the user to the Add Title page

5.1.13 Add Title

Add Title

New TitleID is automatically generated.
5.1.14  Edit Title

Edit Title

[Diagram of an interface for editing titles with fields for name, description, minimum and maximum base salary, bonus allowed, minimum and maximum bonus.

5.1.15  View Title

View Title

[Diagram of an interface for viewing titles similar to the Edit Title interface.

---
6 IMPLEMENTATION GUIDELINES

This SOA Specification has been designed to be platform agnostic. The underlying services, database integration, Web applications and portal functionality should be implementable on a variety of commercial and open source platforms.

The database schemas provided are for Oracle and SQL Server databases to act as a basis for J2EE and .NET implementations, respectively. It is acceptable to create schema on other databases providing the given entities and their relationships are retained. It is not permitted to create stored procedures for any database.

The business and workflow services specified must adhere exactly to the given interface specification. Where WSDL is provided this must act as a basis for the service provider and consumer implementation. All business services should follow the process flow defined in their flow diagrams. It is required that concurrency, exception handling, email notification and all other constructs are included in the specified process.

Where a particular transport is specified that must also be adhered to. It is unacceptable to implement a Message Broker driven service over http for example. Security must also be applied as specified where required.

Web applications should follow the guideline given. It is required that the given pages are produced and the page flows be followed. It is acceptable to include Web actions as controllers between pages to manage the flow. All specified fields and functionality should appear on the pages.

Portal applications should also follow the given guidelines. Again, the portlets and portlet flow should be adhered to, all specified fields included and portal functionality such as sign in, minimize, maximize, etc. included.
APPENDICES

A. Schema for common databases
   To be included in file format in implementation kit.

B. WSDL for services
   To be included in file format in implementation kit.