Project	P02439 – Business Service Delivery
	Non-Functional Requirements, Service Characteristics and Policies
Version	1.1

Date: [18/10/07]

Table of Contents

1 Introduction

The Business Service Layer provides a set of services to satisfy specific business system requirements. The services are provided as Web Service operations which, in principle, can be reused by disparate consumers. The services and operations are defined in the relevant service contract. Consumers must satisfy themselves that the service meets their requirements and can work within any constraints imposed by the service; they must also be authorised to use service operations by the service provider.

This document is a constituent part of the service contracts and defines non-functional requirements, characteristics and policies which apply generally to all services in the Business Service Layer. In some cases, such as performance, these can only sensibly be defined at the service and operation level in which case those details will be included in the contract definitions.

2 Service Characteristics

This section describes characteristics and policies which apply to all Business Services. There will be cases, such as specific performance, latency and response time requirements which must be defined at the individual operation level.

2.1 Policies

2.1.1 CHANGE CONTROL

The change cycle of Services is generally independent of that of consumers. It is expected that Business Services will change without recourse to clients.

Where a breaking change is identified, be it functional or physical, it will result in a version change. Versioning will be applied as described in §2.1.5.

2.1.2 RETENTION

Unless otherwise specifically agreed with a consumer, a previous version of a service will be retained in operation with a target decommissioning date of six months from the date it ceases to be the most recent version. Only one previous version of a service will be retained; should three releases occur within a six month period the oldest version of the service will be decommissioned ahead of the original decommissioning date.

2.1.3 DECOMMISSIONING

When a version of a service reaches its end of life, consumers of that service will be informed that it has been scheduled for decommissioning. The date of decommissioning will be six calendar months after the version of the service ceases to be the "most recent".

The date of decommissioning may not exactly correspond to this but it will not be earlier. Where consumers are not able to migrate to a new version of a service within the specified timeframe consideration will be given to extending the active period of that version. It should be understood that this will be an exceptional occurrence.

2.1.4 ACCESS

Publicly-available services offered by the Business Service Layer will be as WS-I Web Services via the SOAP protocol.

2.1.5 VERSIONING

Versioning of the BPM Service interfaces will follow Application Engineering's recommendation to use the UBL approach[*] applied to XML Schema Definition (XSD) and Web Service Definition Language (WSDL) filenames and their defined namespaces[†] and to service end-points.

Major version numbers represent "breaking" changes to a service. Breaking functional changes occur where the processing logic of the service implementation changes to such an extent that the operation(s) no longer fulfil the same functional contract as previously agreed. Breaking physical changes occur where the interface of a specific operation changes. The changes are such that a consumer will need to change their implementation and adopt the service change in order to continue

[*] http://www.idealliance.org/papers/dx_xmle03/papers/03-04-03/03-04-03.html

[†]

using it. Typically breaking physical changes would be alterations to the interface, or deletions from it. It is expected that both types of breaking change can occur on the same revision.

Minor version numbers represent a change to the interface which will allow existing consumers of a service to continue without change and without affecting their usage; typically this would be where additions or extensions are made to an interface.

Public versioning of schemas will be two-level, major and minor numbering; versioning of WSDL and end-points will be single level indicating major version only. Filename and namespace version numbers must indicate the same version.

2.1.5.1 SCHEMA VERSIONING

In practice the ABC service schema will be defined in the ABC_*X*_*Y*.xsd file(s) in the appropriate location in the project structure; *X* is the major version, *Y* is the minor version. Within the schema definition, the following convention will be used for versioning the namespace:

http://xmlns.fid-intl.com/bsd/*servicename*/*part*/*X*.*Y*

where *servicename* identifies the service component of the namespace, *part* identifies a significant sub-part to the namespace, eg. enterprise data type definitions, and *X* and *Y* are as above. By convention, all elements of the namespace are lower-case. For example, the following is the definition of the Charges service interface namespace:

http://xmlns.fid-intl.com/bsd/charges/chargesservice/1.0

The minor version is incremented when a non-breaking change is made to the relevant schema definition; the major version is incremented on a breaking change to the service.

2.1.5.2 WSDL VERSIONING

WSDL namespaces will be versioned at the major level only. The following model will be used:

http://www.fid-intl.com/bsd/service/*servicename*/*X*

where *X* has the same meaning as above. For example, the following allow access to the version 1 WSDL of the Client service:

http://www.fid-intl.com/bsd/service/clientservice/1

2.1.5.3 PORTTYPE VERSIONING

PortTypes within the WSDL will be versioned at major.minor level according to the following:

<wsdl:portType name="*Service*WebService_*X*.*Y*">

where X and Y have the same meanings as above.

The following is an example for version 1.1 of the Client service:

<wsdl:portType name="ClientWebService_1.1">

2.1.5.4 BINDING VERSIONING

In line with portTypes, bindings will also be versioned as major.minor according to the following:

<wsdl:binding name="*ServiceName_X.Y*_SOAP11Binding" type="*portType*">

where X, Y and ServiceName have the same meanings as above, *portType* refers to the associated portType.

The following is an example for version 1.1 of the Client service:

<wsdl:binding name="ClientService_1.1_SOAP11Binding" type="tns:ClientWebService_1.1">

2.1.5.5 SERVICE VERSIONING

Service definitions will also be versioned with major.minor according to the following:

<wsdl:service name="*ServiceName_X.Y*">

where X and Y have the same meanings as above.

The following is an example for version 1.1 of the Client service:

<wsdl:service name="ClientService_1.1">

2.1.5.6 END-POINT VERSIONING

Service operation end-points will also be major-only versioned as above. The following model will be used:

http://www.fid-intl.com/bsd/service/*servicename_X*

where *X* has the same meaning as in schema versions. For example, the following would be the end-point for major version 1 of the Client service:

http://www.fid-intl.com/bsd/service/client_1

2.1.6 SECURITY

The security model for all Business Services is shown in Figure 1.

Wherever a password is retained it will be in an encrypted state.

The security at the various levels is:

A. Transport is 2-way SSL, bilateral certificate exchange; basic authentication is applied application-to-application. Using SSL, asymmetric encryption is used to establish the conversation, symmetric encryption is used thereafter.

B. Transport is .NET binary using Windows-level authorisation; authentication requires that all BizTalk applications belong to a restricted BizTalk user group.

C. Transport is 1-way SSL; basic authentication is applied at the application level, authorisation is role-based at the level of the consuming business service.

D. Transport is 2-way SSL, bilateral certificate exchange; basic authentication is applied at the application level, authorisation is role-based.

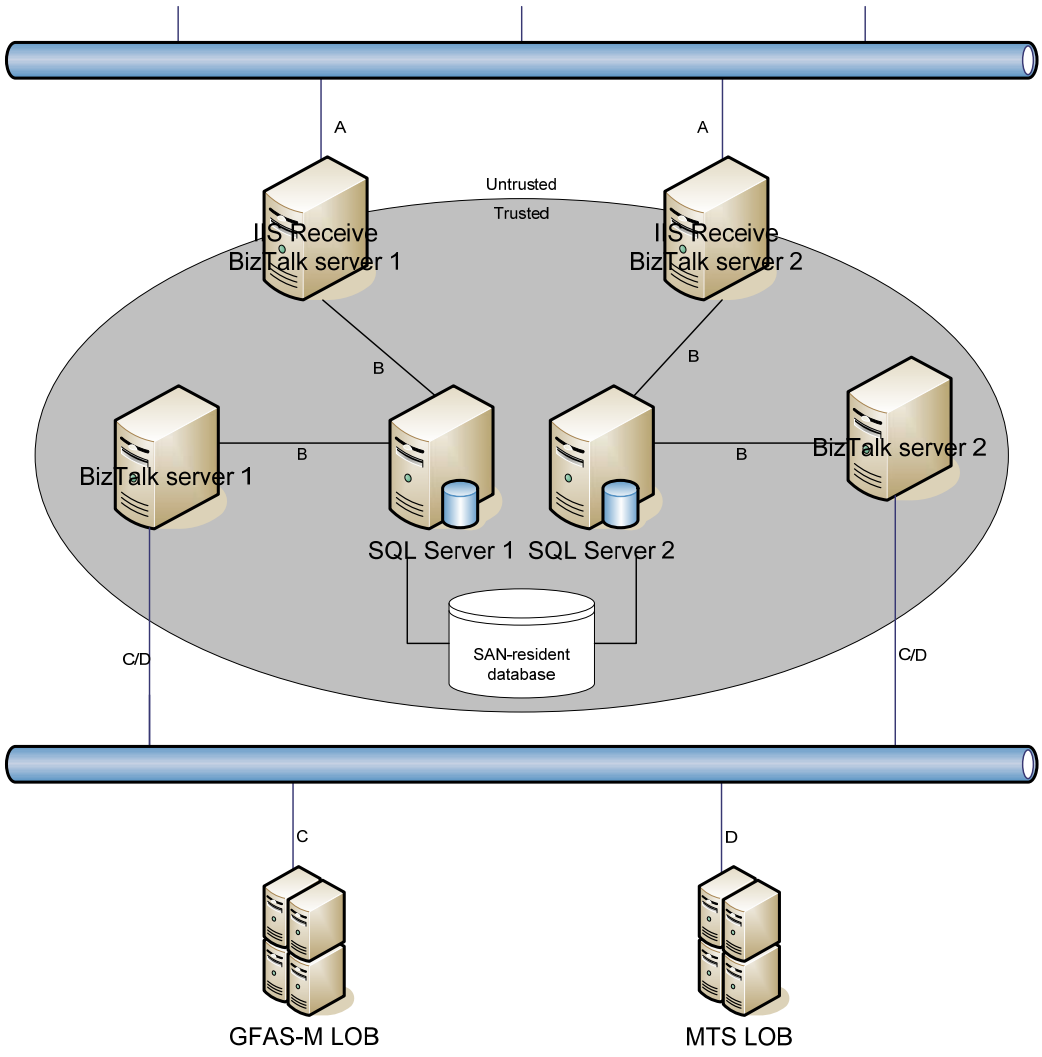

Figure 1: Security Model

2.1.6.1 AUTHENTICATION

Authentication applies on each transit point on Figure 1 as follows:

A. Basic authentication is applied to consumers based on the certificate exchange. Each request is separately authenticated.

B. All participants at this level must belong to the local BizTalk user group. Windows-level authorisation is applied.

C. Basic authentication is applied at the application-to-application level.

D. Basic authentication is applied at the application-to-application level.

All authentication activity will be logged at the point of the authentication attempt.

2.1.6.2 AUTHORISATION

Authorisation applies on each transit point on Figure 1 as follows:

A. Authorisation is at the consuming application to service operation level.

B. Application-level authorisation applies to the BizTalk user which is authorised to use SQL Server services.

C. Role-based authorisation applies to the consuming business service.

D. Application-level authorisation applies; the consuming application must be recognised in a specific user group on MTS.

Each application is responsible for implementing its own authorisation mechanism. In the case of the BizTalk domain this will be implemented via policies installed within IIS.

2.1.6.3 ENCRYPTION

All messages into and out of the Business Service Layer are encrypted by the use of SSL; that is, all traffic on legs A, C and D is encrypted. Within the trusted domain traffic is .NET binary.

The BizTalk domain's Single Sign-On (SSO) service has the capability to manage encryption and decryption of sensitive data and it will used to store such data as debit/credit card numbers before messages are stored in the message box.

2.1.7 SERVICE LEVEL AGREEMENT

This section describes the Service Level Agreement (SLA) applicable to the Address Service generally.

2.1.7.1 AVAILABILITY

Business Services are AAA-rated so have a maximum recovery period of four hours.

The infrastructure topology has two paired servers, each acting in active/active configuration ready to fail-over in the event of a system failure. One pair act as the receive servers running the IIS service and the associated BizTalk receive installation. The other pair act as process servers and run the bulk of the BizTalk processing.

The SQL Server cluster will operate in active/active mode and will be able to switch load in the event of one system failure. It will also host the SSO Master service which will also fail-over on failure of one system.

Both back-end application service host environments are also AAA-rated and have full disaster recovery implementations.

2.1.7.2 PERFORMANCE

The overall performance requirement for the BPM project as a whole is to be able to process 41 500 new business applications per month. To establish the hourly volume:

1) The working day runs from 6am to 12 noon, ie. 6 hours.

2) Compensating for weekends, a working month is (365 – (52 x 2)) / 12 = 21.75 days.

3) The working month is equivalent to 21.75 x 6 = 130.5 hours.

4) The hourly volume is therefore 41 500 / 130.5 = 318 transactions per hour.

The suggestion has been made to test simulated peak volumes modelled on 90% of the overall volume serviced in 10% of the time (the "90-10" peak). Based on this:

5) 90% of overall volume is 41 500 x 0.9 = 37 350.

6) 10% of the monthly working time is 130.5 x 0.1 = 13.05 hours.

7) The simulated 90-10 peak volume is therefore 37 350 / 13.05 = 2862 transactions per hour.

From 4) and 7) above, the mean (and peak) volumes are 318 (2862) transactions per hour. This is equivalent to 0.09 (0.8) transactions per second, or about 11.32 (1.25) seconds per transaction.

The figure of 41 500 messages per month is published in the Business' SRA and it has not been possible to refine it or gain any further certainty over it. If a new figure is published it will obviously affect the performance requirements. There are no figures currently available indicating how the number of service calls relate to transactions or how the volume of validation resubmissions relate to an individual transaction.

2.1.7.3 RESPONSIVENESS

Where operations use the request/response (synchronous) message exchange pattern, a timeout will apply to all requests; these will be defined at the operation level in contracts once the performance profile is known. Consequently, service consumers will not block indefinitely waiting for a response.

There are three levels of timeout:

- Service-side operation-level SOAP timeouts. These will be configured on the send-ports; they must be defined relative to operation implemented delay monitoring.
- Service-side operation implemented delay monitoring. Many operations are iterative and multi-threaded and to prevent indefinite waiting on resources, for instance, maximum wait periods and poll counts will be defined per operation; where these apply they will be configured using the application configuration file(s). These events will be indicated in the operation error block.
- Client-side SOAP timeout. The consumer must set operation timeouts to be *greater* than service-side operation timeouts. In the event of consumers setting timeout values to be too short with respect to the operation-defined timeout and that timeout firing, then undelivered response messages will build up within the Business Services infrastructure. These can be reported using MOM and the BizTalk Administration Console. This would also apply should a consumer (somehow) call a synchronous operation asynchronously.

Timeouts will not apply to request/notify (asynchronous) message exchanges.