ISB Enterprise Canonical Data Architecture Design Methodology

Version: 4.0

Issue Date: 04/08/2014

Document Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Issue Date</th>
<th>Modified by</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Draft</td>
<td>04/06/2014</td>
<td>Stephen Palmer</td>
<td>Publication for internal review</td>
</tr>
<tr>
<td>2.0</td>
<td>Draft</td>
<td>10/06/2014</td>
<td>Jo Bowers</td>
<td>2\textsuperscript{nd} Internal Release</td>
</tr>
<tr>
<td>3.0</td>
<td>Draft</td>
<td>12/06/2014</td>
<td>Jo Bowers</td>
<td>3\textsuperscript{rd} Internal Release</td>
</tr>
<tr>
<td>4.0</td>
<td>Final</td>
<td>04/08/2014</td>
<td>Amanda Hopewell-Campion</td>
<td>Initial External Release</td>
</tr>
</tbody>
</table>
Contents

1 Introduction .......................................................................................................................... 3

2 Problem definition .............................................................................................................. 3

3 Data Architectural approach ................................................................................................. 4
   3.1 Overview of the approach .............................................................................................. 4
   3.2 Architectural methodology ............................................................................................ 5
       3.2.1 Key benefits of this approach .............................................................................. 7
   3.3 Overview of the data architecture model ....................................................................... 7
   3.4 Encoding the Canonical Model for ISB ....................................................................... 8
   3.5 How transition (non-ISB) data blocks are defined ....................................................... 9
   3.6 How data blocks operate (as building blocks in a business/message payload) ............. 9
   3.7 Generic options for determining data to extract .......................................................... 11
       3.7.1 Attribute level change extraction ....................................................................... 12
       3.7.2 Table level change extraction .............................................................................. 12
       3.7.3 Difference engine extraction .............................................................................. 12
       3.7.4 Full data extraction ............................................................................................. 12
   3.8 Frequency of data extracts ............................................................................................ 13
       3.8.1 Real time data extracts ....................................................................................... 13
       3.8.2 Timed data extracts ............................................................................................. 13
   3.9 Using Business to Business Transactions ..................................................................... 14
   3.10 Developing a common Application Interface .............................................................. 15
   3.11 Data Quality and Data Ownership .............................................................................. 18
   3.12 Master data and reference data in the enterprise ......................................................... 18

4 Glossary ................................................................................................................................. 19

5 Copyrights ............................................................................................................................. 21
1 Introduction

Future-proofing and optimising enterprise data exchanges requires a unique approach.

Traditional methods define a set of interfaces suitable for the particular data exchange needs at the time. This results in a proliferation of data formats and interfaces that require constant management and alteration. The net effect is that the data exchange becomes a heavy financial burden and constraint on growth in an enterprise.

The alternative approach requires data and information to become the focus of the data exchange, thereby enabling future-proof real-time reusable data exchanges. This reduces the number of interfaces required, maintenance costs and enables growth in the enterprise. This approach requires an interoperability solution focusing on the data layer. This key delivery component can then be supplied across many different transport layers for different data exchange needs at different times.

2 Problem definition

Whilst there are many areas of system design that can be scrutinised, one of the core issues that is very rarely recognised is the approach to data.

Traditionally, many systems are built on a solution based data design. In principle this means the focus is on the data required to answer the question that that system is intended to support. This leads to an un-optimised design of data with the data either being derived or designed specifically for that solution.

Worse than this is the tendency to then develop additional solutions by using existing data from other solutions. On the basis that this design approach has already “compromised” the data, to further use it for a solution that the data was never intended for will lead to data issues.

Many interoperability solutions are also designed on the above premise to exchange data. They tend to be large batch based designs intended for a specific purpose. Each time new data is required to be exchanged a new interface is defined leading to a proliferation of interfaces that need to be maintained. Ultimately, the “same” data will appear in multiple interfaces but when reconciled, it will be discovered that the contents of the data differs. This will be due to each interface adding subtly different data extraction rules to serve the different business processes for which it was intended.

The result of this approach is that business flexibility is reduced, interfaces are complex, costly and time consuming to maintain and that adding new systems or new data is a major undertaking.
3 Data Architectural approach

3.1 Overview of the approach

In reality, an organisation’s business model relies on what can be described as the atomic level business data. This is the data that is involved in the business activities and processes such as the lifecycle of the customer, client, products etc.

Therefore, the correct future-proof solution requires a radical rethink of data in these terms. There is already a term for this concept and that is the Enterprise Data Architecture (EDA). This concept is realised by the ISB through the document called the ISB Enterprise Business Data Architecture. ISB-000108

The objective is to define a Data Architecture for the Enterprise, rather than the system. The intention is to simplify and future-proof the exchange of data.

The implementation of this methodology relies on the experience and understanding...
of the business to be able to define the atomic level business data that will form the EDA.

Enterprise Data Architecture does have a tendency to be considered as a conceptual approach only and not as a solution for the physical implementation. It is certainly a logical design for the data but can equally be used as a design for the physical implementation. There may be reasons why the physical design might need to be optimised further but the realisation of the solution should be also be based on the EDA. The risk of changing the physical implementation of the solution is that ultimately it does not behave in the same way as the Enterprise Data Architecture design and causes more problems to be introduced.

It is important to understand the following methodology, supporting this EDA design, to ensure implementers can easily translate the logical to suitable physical implementations.

### 3.2 Architectural methodology

The methodology required is a bespoke design based on Enterprise Data Architecture methodology and Database design, but is a methodology that is fully integrated from the conceptual design through to the implementable solution. The resulting design can then be implemented as-is as an interoperability solution. It also becomes the canonical data model against which the organisation can decree all of their systems must adhere at a semantic and behavioural level.

The Canonical Model is the standard data model for the enterprise and is a generic data model that can be plugged into any system without any dependency on the applications used. Instead of translating data between each and every application, it is then sufficient just to translate data between each format and the canonical format.

The methodology consists of the following activities:

- Start by identifying and modelling the logical business processes that are involved in the organisation and the data involved at step of each process.

- Identify all of the data involved across all of the business processes and decouple the data from the processes to detail the pure atomic data of the real enterprise.

- Construct a decomposition Entity Relationship Diagram (ERD) which has three levels: Domain Model, Level 1 Entity Model and Level 2 Fully Attributed Model. Each level decomposes to the level below; they are NOT separate models. This creates a data model that groups the atomic data into entities based on their common behaviour and models the relationships between those entities.

- A strict methodology is applied, which is a balance between an object model and a physical model, that results in a fully attributed ERD model with, where
possible, natural identifiers. That is, identifiers that form the primary key of each entity created from the real business data rather than creating any artificial identifiers or keys. The balance is to ensure re-usability without being too esoteric. The methodology includes rules about sub-typing, identifiers, full lifecycle entities amongst many other things. The result of this is the future-proof enterprise **canonical model**.

- The Canonical Model may need to use terms that are more abstract than used in the business processes. This is because one entity in the Canonical Model can be used to support several different business processes. Therefore the attributes in the Canonical Model are then mapped back to the data in the business processes to ensure that that each data item is covered in the Canonical Model and it also shows the high re-usability of the resulting canonical model.

- The Canonical Model is then broken up into distinct areas of related data and these become the ISB data blocks used in the exchange of data. One data block may contain one, or several related entities from the ERD. Each data block is described completely by one ISB Business Data Standard. These data blocks are highly re-useable and fit together similar to building blocks to describe any business process required. This is then the system solution sought at a particular time; the building blocks are the defined-once-used-many data structures.
Illustration of the EDA methodology where interfaces are standardised with the Canonical Data Model. The equivalent Data Collections are found by combining different ISB data blocks

3.2.1 Key benefits of this approach

- Data is structured in a standard format.
- The meaning of each piece of data is standard across the Enterprise.
- Data reflects the processes in use by the business, not the view of the originating system. This eliminates system “lock in” where data cannot be easily changed because it has been designed for that system.
- New data of a type already exchanged, such as a new identifier for a person, does not require changes to the data exchange.
- Data is exchanged in its atomic format making it re-useable.
- All data, such as a person’s details, are sent only once from a source and is reused as required in any business process.
- Data can be provided in near real time, dramatically reducing data volumes and providing up to date information allowing the business to respond rapidly.
- All data exchanges use a single standardised format.
- Additional data requirements can be easily added at minimal cost/time.
- Existing data is utilised to identify data blocks through the use of natural identifiers.
- No change is required to the internal data structures of source or destination systems.

3.3 Overview of the data architecture model

Data in different systems rarely match in terms of name, format, structure etc. However, all data in collaborating systems must have similarities to be interoperable as they all support the same business processes. To be truly different, one system would have to support, say, a book store and the other support medical health care. The issue would be that there is no common “currency” that allows different currencies of data to be triangulated through. A common currency allows a conversion from one currency to another, such as going from pounds to euros via the dollar. The dollar is a common currency and, using exchange rates and currency can be converted.

The future-proof Canonical Model is the “common currency” in the data exchange making different systems interoperable. Using the future-proof Canonical Model, data is converted at extract from a system into the canonical format and upon receipt is converted into the internal system format. All system exchanges are able to conform to this one design regardless of their internal system design.

The enterprise now has one data exchange design, dramatically reducing the complexity, maintenance costs and implementation time for new systems or changes.
to the business. Implementing new systems becomes plug-n-play.

The Canonical Model has been broken up into distinct areas of related data to define the data blocks used in the exchange of data.

The data blocks can then be converted into a physical data exchange encoding such as XML. Each XML encoding of a data block is described completely in an ISB Technical Data Standard. The conversion includes a strict set of rules for how to convert the data blocks from the ERD into an XSD such that it mirrors the behaviour defined in the ERD.

The resulting encoded data design can now be used on any transport medium with the confidence that every single system that exchanges data in this format will be compatible and the resulting data received will be re-useable.

Interoperability solutions can be batch or “real time” using the same data design and therefore can be migrated from batch to “real time” without altering the design of the interface, just the frequency of exchange.

Systems can implement the data design directly into their database, providing all of the benefits, should they wish to. Otherwise they will need to transform the data to the canonical data format on the extract of data in order to exchange data with other systems.

### 3.4 Encoding the Canonical Model for ISB

As mentioned above, the Canonical Model can be encoded in many ways such as XML, CSV, or a variety of other methods.

The method for ISB for exchanging information is XML for the following reasons:
- XML can easily represent the hierarchical nature of the Entity Relationship model and also through hierarchical structuring reduce data redundancy
- Is a widely used encoding with a solid set of standards supporting it
- With the use of clear tags the data format in a message can be easily interpreted by a human.

**Version 5.0 of the Achieving Conformity ISB-000188 Document – Appendix A** states:

**Technical Data Standard (TDS):** an ISB produced TDS defines a ‘data block’ for its corresponding BDS using the XML encoding format. As such, it is ready to be ‘dropped into’ a XML based Application Interface Specification wherever the data block defined by the TDS is needed.
3.5 How transition (non-ISB) data blocks are defined

The Enterprise Data Architecture is one that evolves over time and the Canonical Model may not immediately encapsulate all of the data in the enterprise or may not immediately encapsulate entirely new business processes as they are added to the enterprise.

For these reasons it may be necessary to define data blocks that can carry transitional data that is not conformant with any ISB Business Data Standard, because no ISB BDS currently exists for that data, but which can still be exchanged in a way that is consistent with the EDA. These are termed transition data blocks as they a temporary solution to this problem.

A transition data block will carry transitional data in its existing format which has not been normalised with the EDA and therefore does not provide the same benefits as the EDA. However, in designing a transition data block the transitional data must be encapsulated in a way that makes it possible to create relationships with the foundation entities in the EDA so that this data can be related to data in the Canonical Model.

This is done by extracting the EDA foundation entity identifiers from the transition block and then creating a primary key for the transition data block using the extracted foundation entity identifier(s) plus a surrogate key. Ideally, the full natural identifier would be extrapolated from the transition block but the behaviour of a transition block is likely to be a de-normalised design and as such not suitable for a natural identifier as defined by data architecture design methodology.

For this reason, any transition data blocks must still be defined by the Enterprise Data Architect to ensure a consistent approach. Once the transitional data is subject to the full data architecture design methodology, the transition block data will be split into a number of data standards, those data standards will be fully incorporated into the Enterprise Data Architecture and the transition data block’s use will be discontinued.

3.6 How data blocks operate (as building blocks in a business/message payload)

The output of this methodology is a set of reusable data blocks. However, these may at first appear far removed from the particular problem, such as wanting to exchange a person’s names and addresses and an Organisation’s name and addresses between two systems. Separately the systems might want to exchange individual data fields as they change between the two systems.

Traditionally methods would create a number of interface designs to accommodate the potential transactions:

1) For a new person’s names and addresses
2) For a new organisation’s names and addresses
3) For an amended person’s names and addresses

4) For an amended organisation’s names and addresses

Depending on how the interfaces are developed, it is likely that the system utilises a separate person new data interface a separate changed data interface, a separate organisation new data interface and a separate organisation changed data interface. This very much depends upon how the interfaces develop over time or whether they are architected in advance.

If we then consider the data involved in each of the above transactions we would see that the person name interface will consist of:

1) Person “Full” name
2) Person “Known as” name
3) Person addresses

Likewise the Organisation name interface will contain:

1) Organisation “Incorporated” name
2) Organisation addresses

On inspection, it can be seen that the person information is the same type as the organisation information but is just the data for a person rather than for an organisation.

The above is beginning to identify that rather than creating individual specific data interfaces, it is possible to define a canonical data set such as:

1) Person/Organisation names
2) Person/Organisation addresses

We can create a set of highly re-useable data blocks that, if we wish to support the original interface requirement of a person’s names and addresses and an Organisation’s name and addresses included, instead of a number of very specific interfaces we can just send the following data blocks:

1) Party Name
2) Locator
3) Party Contact

So the only difference between this approach and our traditional interface design is that we exchange smaller units of data, the data blocks, which “add up” to the total data required for these exchanges.

When considering data exchange between two systems the traditional method is to create a data extract for each Business to Business transaction and then embed all of the data required to support that transaction on the basis that none of the will data already exist in the receiving system.

This approach can be used but is not optimal and will result in data being exchanged that doesn’t need to be. A better approach from the aspect of data volumes and integration is exchanging data at the data block level when it changes in the source system. This is known as a Data Driven approach.
This Data Driven approach does rely on close synchronisation of the source and receiving systems. This approach works well for a receiving system that requires all data from the source system, such as an Operational Data Store.

For systems that only want the data needed for a business process there will be a **Transaction Driven** approach. This is when a pre-defined set of data blocks is extracted from the system as a result of a Trigger event which can either be a process or a change in data. The data that is extracted is treated as a single Business to Business Transaction and is processed accordingly.

One example is a Learner being booked on a Qualification Assessment. The Trigger Event is the Learner Booking process which then sends all of the Learner details along with the booking to the Awarding Organisation as a single Transaction. In the event that the recipient Awarding Organisation checks the data before loading and identifies that the Learner should not have been booked on that assessment, they would reject the whole Transaction which would include the Learner Booking and all the Learner details.

Therefore, the primary difference between Data Driven and Business to Business Transaction Driven is:

- **Data Driven**
  - Monitors data and only extracts the data that has changed or is required within a single data block
  - A message can have multiple data blocks and each one will be treated independently

- **Transaction Driven**
  - Data is extracted as a result of a Trigger Event which can be a business process of monitored data
  - A pre-defined set of data blocks is extracted for a transaction which may be much more that the data that has changed as a result of the Trigger Event
  - The whole Transaction is treated as a single Business to Business Transaction and either fully processed or fully rejected. The receiver must not process individual data blocks into their system until the whole Transaction has been checked and all data blocks could be accepted.

Implementers will now be aware that ISB data blocks can be used to exchange data in a number of ways. Each method needs consideration based on the systems designs and business processes. The next section discusses in more detail data monitoring options that will help with identifying the particular end to end solution required.

**3.7 Generic options for determining data to extract**
Whether Data Driven or Business to Business Transaction Driven (Data Event Triggered) data exchange requires that the system database is monitored and whenever a change to a record occurs the system can, immediately or at predetermined times, extract those changes and send them to any other interested system (as long as the receiving end-point is permitted to see the data).

The frequency of data exchanges is determined by the time window (i.e. latency) of the data extract. That is, a time window will be specified that states that all changes to data must be sent via data exchange within, say, 24 hours of the change occurring. Within this time window individual systems can decide whether to send changes immediately or whether to send a number of changes together at predetermined times within the data extraction window.

The sending system will need to maintain a control table that records the last extracted date/time for each change that has been sent. The system therefore needs to monitor the database for any changes to data since the last extracted date/time for that data.

As systems are unlikely to structure their database at the same level of granularity as the data blocks there may be one data record in the sending system that equates to multiple data blocks in the canonical model. This would simply result in multiple data blocks being created for the one “change” extracted from the system.

The level of monitoring for change can vary between systems. The following sections consider the options available

### 3.7.1 Attribute level change extraction

A sending system that records changes to data at an attribute level will record an updated date/time per attribute. The system will be able to extract the individual attributes that have changed (i.e. have an updated date/time) since the last data extract window.

### 3.7.2 Table level change extraction

A sending system that records changes to data at a record level will have an updated date/time per record. This will require the system to extract the entire record whenever any attribute changes on that record.

### 3.7.3 Difference engine extraction

Sending systems might employ a ‘difference engine’ to compare data before and after a particular data extraction window; data is compared against a ‘shadow’ copy of the data to identify attribute level changes that have occurred.

### 3.7.4 Full data extraction

Some sending systems might be built with no change capability, i.e. no use of an updated date/time, which implies that all data in the system must be sent at every
data extract window because there is no way being any more precise than this. Due to the potential for large volumes of data being exchanged this approach could only be employed by systems that supply very small, fixed ranges of data.

### 3.8 Frequency of data extracts

Data can be extracted from the system as the monitored change to the data occurs or on a scheduled basis.

Each implementation needs to determine which changes to data must be extracted in real time and which data should be timed extracts. The reason for this is that whilst real time data extracts can appear ideal, if the amount of data that would be extracted as a result of the real time extract was large it may impact the systems on-line performance. Therefore, large data volume extracts should, where possible, be run “out of hours” of the on-line usage of the system. This is likely to be a balance between the importance of the data and the volume of data or messages that are likely to be created as a result.

The following outlines the two options.

#### 3.8.1 Real time data extracts

Real time data extracts would extract data and exchange it whenever a change in the data is detected. This would result in a message being created that contains only those changes that have been detected at that time.

#### 3.8.2 Timed data extracts

Timed Data Driven extracts would extract data according to a schedule or frequency, such as once per day. This would result in a message being created that contains all of the changes that have been detected since the last timed data extract was executed.

To implement this approach the system needs to be able to detect all the data that has changed since the last timed data extract was executed. This can be achieved through the use of a system Control Table, either by:

- i. Recording the last data extract execution date/time in the Control Table and querying the internal database for updated date/times that have occurred since the last extract date/time
- ii. Using the Control Table to record the actual changes to data within the system as they occur, along with the date/time of each change, and then querying the Control Table for all changes that have occurred since the extract date/time

The advantage of this approach is that many data changes can be batched into
messages and sent at a predetermined frequency. This results in fewer messages being exchanged and messages which can be exchanged in a more managed way. The trade-off is that changes will not be seen by the receiving system in real time but at some later time.

3.9 Using Business to Business Transactions

Business to Business Transactions occur where one system requires a data extract from another system to support a business process regardless of whether all of the required source data has changed or not. This transaction is usually invoked in response to a Trigger Event which can either be a process being run or a data change. One example is a Learner being booked on a Qualification Assessment. The Trigger Event is the Learner Booking process which then sends all of the Learner details along with the booking to the Awarding Organisation as a single Transaction.

One of the common mistakes in Business to Business exchanges is to reject interface records because the validation rules applied by the receiving system are overly aggressive.

Business to Business by its definition is about collaboration between systems. Over aggressive validation can result in a record being rejected by the receiving system even though the sending system accepted the data as valid. The rejection then results in:

a) The sending system having to be notified about the error and instigating manual processes to resolve the “perceived” data error.
b) The sending system having to resend data that was previously sent by re-flagging that data for extraction.
c) The receiving system being unable to carry out other processes until it receives a valid data extract, even if some parts of the original data exchange were valid.

With the use of the data block concept, the occurrences of (c) are reduced as each data block is independent and can be validated in isolation.

To refer to an earlier example, the single interface that contains both pupil details and assessment details would be rejected if the assessment data was invalid even if the pupil details were valid. Using the data block method the pupil details would be separate and only the assessment would be rejected.

The intention of the validation and the Business to Business Transaction however, needs to be closely considered. Should the receiving system only be notified when there is a pupil with a valid assessment, otherwise it is redundant information, or should the receiving system be notified that it needs to be aware of a pupil, for good business reasons, even if no valid assessment can be sent at that time?

The key point here is NOT to use the interface as a validation service for business rules that should be embedded in the actual business processes and systems themselves.
If the data exchange solution architect considers that the interface is a “pipe line” between systems to integrate them, the purpose of the interface design is to:

1) Provide a common format of data that both systems can recognise.
2) Ensure that any data entered into the interface meets the field format and if there is an issue it immediately informs the sending system (i.e. length and data format only).
3) Assist Business to Business Transactions by not adding business level validation that can change and should be built into the sending or receiving systems themselves as that same validation will apply to data directly entered into those systems via data entry screens.
4) Reject data received via the interface only if it fundamentally would fail any process the receiving system attempted to enact on the data such as an assessment being received for a pupil who does not exist. Generally this will be dealt with through referential integrity rules.

An important aspect of the Data Driven versus Business to Business Transaction approach is that the interface is only part of the end to end Business to Business Transaction. Every Business to Business exchange has to be designed end to end so that its purpose is clearly understood and managed. Just sending data from system A to system B that then gets rejected from system B without having the correct feedback processes, system alerts and administration reports will result in any follow up queries coming to a dead end.

The only difference between a traditional solutions based interface and the data block approach is that there will be multiple data records exchanged for the data block approach versus one record for the solution interface design. The actual data sent in both methods will be the same.

3.10 Developing a common Application Interface

By defining that each data block can be both outbound and inbound, it is possible to invoke each data block through a number of common interface components.

Firstly, it is possible to construct a common error message handler to manage any errors when loading data into a data block.

Secondly, it is possible to design an application interface (API) for each data block that:

a) Contains a set of variables for each attribute in that data block
b) Maps those variables to the appropriate attributes in the data block
c) Calls the common error handler if there is an issue when loading data
d) Creates the XML record for that data block in an interface file

When extracting data the system will use these common components in the following way by:

a) creating an application that runs for a specific set of data, such as pupil details
b) identifying the particular set of data block(s) that support that data
c) For each data block identified:
   i. extract the data from the internal database and load each data item to
      the variables as defined in the API for that data block
   ii. call the particular data block API for that data block to create the XML
       record
   iii. process the next data block in the sequence

d) When complete, either process the next data extract or pass the message to
    the transport layer to send the data to the receiving system

When a system receives data, the system will use these common components in the
following way by:

1) Read the XML message to identify the data block(s) contained in the
   message
2) For each data block read:
   a. Process the XML record by passing each XML element to the above
      predefined variable outlined above
   b. Call the data block API for that data block
   c. Attempt to update the data block into the system
   d. If the data does not exist then update the data
   e. If the data already exists then update the data
   f. If the data block is marked as delete then either physically the data in
      the system or mark the data as logically deleted (by updating a record
      delete flag)

Process the next record or if at the end of a data block process the next data block.

The main points to gaining the advantages of the data blocks are to:

1) Create common objects where possible such as error handling routines and
   pass different errors as variables
2) Create each data block as an independent object that can be called as
   required
3) Pass data for a particular extract from the system to the data block as a set of
   variables to decouple the data block from the data being extracted. This
   allows both a School or Pupil to be extracted independently from their
   physical locations in the system but passed to the same data block handler
4) When receiving data use the above techniques in reverse

By using the above design techniques the data block system can be implemented
quickly and simply. Whenever new types of data extracts are required to be
exchanged that use the same data blocks it is only the extract from the system and
passing that data to the variables that needs to be amended. All the other logic uses
existing objects so no further development is required. The above steps may vary
from system to system where, for example, before loading data to the system a user
interface may allow a system administrator to accept or reject the record. With the appropriate system interface design the benefits of using a common set of objects with variations to the above can still be realised.

This design recommendation is also true whether the data exchange requires real time data extracts, timed data extracts, is Data Driven or Business to Business Transactions Driven.
3.11 Data Quality and Data Ownership

An enterprise data architecture requires that source systems be the owners of their own data. This means that source systems are responsible for data quality and data validation.

Any data validation issues are most quickly dealt with upon data entry, where incorrect data can quickly be corrected by the end user. In this way any data exchanged from the source system is correct in any receiving system.

Without this, poor data validation at source would result in receiving systems containing poor quality data that becomes difficult and costly to correct. Feedback messages from the receiving system can request that the source system corrects its data and resends that data. However, unless this is administered in a consistent way there is a risk that different receiving systems with that data will become out of sync.

Worse still would be if a receiving system attempted to correct any data within its own systems as it would then be out of sync with the sending system.

For these reasons data quality must be the responsibility of the sending system.

3.12 Master data and reference data in the enterprise

As part of the Canonical Model development, reference data will be defined. This is a key component in the design of an Enterprise Data Architecture as each system is likely to have its own values for code lists but the enterprise needs to operate consistently. Therefore, the Canonical Model will define a set of reference data standards that reflect the business nature and semantic nature of each code list but harmonises the values used in the data exchange. The code lists at source and destination do not need to change, it is only in the data exchange where they need to convert to the harmonised value.

In a similar manner, the Canonical Model will quickly establish a set of key entities that ideally require centralised management as the entities are master data. These will include people, organisations, products and services. By managing these as master data, it will be possible to establish trends across the enterprise such as products and services purchased and consumed by a single person in any organisation in the enterprise.

Initially the Canonical Model will identify reference data and its harmonised values and also the key entities in the enterprise. Over time the same design will facilitate the propagation of the reference data and master data from a centralised source over the enterprise.
## 4 Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSD</td>
<td>XML schema definition</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>ISB</td>
<td>Information Standards Board</td>
</tr>
<tr>
<td>BDS</td>
<td>Business Data Standard from the ISB</td>
</tr>
<tr>
<td>TDS</td>
<td>Technical Data Standard from the ISB</td>
</tr>
<tr>
<td>Data Block</td>
<td>One block of data as defined in a BDS and described by an XSD</td>
</tr>
<tr>
<td>Record</td>
<td>In this document refers to one record within a data block</td>
</tr>
<tr>
<td>Attribute</td>
<td>One data field within a record</td>
</tr>
<tr>
<td>Message</td>
<td>A Business Data Message, as described in this specification, which is the XML carrying the Data Blocks</td>
</tr>
<tr>
<td>Organisation</td>
<td>A Party (with Party Type = “Organisation”) who sends or receives data via an exchange of messages and can be identified by a Party Id</td>
</tr>
<tr>
<td>Endpoint</td>
<td>The individual system within an organisation responsible for sending a message</td>
</tr>
<tr>
<td>Capability</td>
<td>The ability of an endpoint to send or receive message with specific types of data</td>
</tr>
<tr>
<td>Initiator</td>
<td>The Organisation that created an individual data message</td>
</tr>
<tr>
<td>Recipient</td>
<td>The Organisation that the above data message is intended for</td>
</tr>
<tr>
<td>Sender</td>
<td>The Organisation that sends a message (Data or Feedback)</td>
</tr>
<tr>
<td>Receiver</td>
<td>The Organisation that receives a message (Data or Feedback)</td>
</tr>
<tr>
<td>Data Flow Policy</td>
<td>The specification of how a particular set of data can be exchanged between two organisations. (See Data Exchange User Specification document).</td>
</tr>
<tr>
<td>Latency</td>
<td>The maximum time between a change in a data item and a corresponding message being sent as specified in the applicable Data Flow Policy*</td>
</tr>
<tr>
<td>GUID</td>
<td>Globally Unique Identifier</td>
</tr>
<tr>
<td>Data Driven</td>
<td>Messages that exchanges data with a recipient when any change in monitored data occurs on the source system</td>
</tr>
<tr>
<td>Transaction Driven</td>
<td>Messages that exchanges data with a recipient when any change occurs to a trigger event that requires a pre-defined set of data block types to be extracted and sent and treated as a single Business to Business Transaction and processed accordingly</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Trigger Event</td>
<td>A particular process or change in data within a system that initiates a Business to Business Transaction</td>
</tr>
<tr>
<td>BDM</td>
<td>Business Data Message</td>
</tr>
<tr>
<td>B2B</td>
<td>Business to Business</td>
</tr>
<tr>
<td>ISB TSS</td>
<td>Information Standards Board Technical Support Service</td>
</tr>
</tbody>
</table>
5 Copyrights

© Crown copyright 2014

5.1 The Information Standards Board (ISB) is an advisory body to the Department for Education (DfE) and the Department for Business, Innovation and Skills (BIS). The information it produces is subject to Crown copyright, which is administered by the National Archives.

5.2 The Crown copyright protected information in this document (other than ISB or Departmental logos) may be reproduced free of charge in any format or medium under the terms of the Open Government Licence, available from the National Archives website.

5.3 Any reuse is subject to the material being reproduced accurately and not used in a misleading context. It must be acknowledged as being protected by Crown copyright and the title of the source material must be supplied with the ISB named as the corporate author.

5.4 Authorisation to reproduce any information in this standard which is identified as being the copyright of a third party must be obtained from the copyright holders concerned.

© SJP Project Solutions Ltd

5.5 All Sections excluding those listed below are copyright of SJP Project Solutions Ltd - © SJP Project Solutions Ltd 2014. All rights reserved:

3.5 How transition (non-ISB) data blocks are defined

3.8.1 Real time data extracts

The SJP Project Solutions Ltd copyright protected information in this document may be reproduced free of charge in any format or medium.

5.6 Any reuse is subject to the material being reproduced accurately and not used in a misleading context. It must be acknowledged as being protected by SJP Project Solutions Ltd copyright and the title of the source material must be supplied with the SJP Project Solutions Ltd named as the corporate author.

You may not, except with our express written permission, commercially exploit the SJP Project Solutions Ltd copyright protected information in this document.