LINK DECOMMISSIONING STRATEGY

CT-KIND Project
STATE OF CONNECTICUT, DEPARTMENT OF CHILDREN AND FAMILIES (DCF)
## Contents

Overview .......................................................................................................................... 3

- Background ..................................................................................................................... 3

- Strangler Pattern ............................................................................................................. 3

- Goals ............................................................................................................................... 3

- Minimize Change to Monolith ....................................................................................... 4

- Minimize Creating More Dependencies ....................................................................... 5

- Create Business Defined Strangler Scope .................................................................... 5

Replacement Strategy ...................................................................................................... 5

Methodologies/Tools .......................................................................................................... 6

- Enterprise Service Bus (ESB) ......................................................................................... 6

  - What is ESB? ................................................................................................................. 6

  - What problems does ESB solve? Why use ESB? ......................................................... 6

  - What is guaranteed message delivery? How does it work? ....................................... 6

  - How can the ESB be used to solve our conversion/migration needs? ....................... 7

- Extract Transform Load (ETL) .................................................................................. 7

  - What is ETL? ............................................................................................................... 7

  - What are some ETL tools that we can use? ............................................................... 8

  - How do you identify the data to be converted and loaded into CT-KIND? ............... 8

  - Just because it can be converted doesn’t necessarily mean it should? .................... 8

  - How do you handle the differences between the data models? .............................. 9

- Robotic Process Automation (RPA) ............................................................................. 9

  - What is RPA? .............................................................................................................. 9

  - How can this technology help us? ............................................................................ 9

Security ............................................................................................................................... 9

- What is the role of security in the migration strategy? ................................................. 9

PowerBuilder Web Services .......................................................................................... 11

- What are web services? REST vs SOAP? What are PowerBuilder specific web services? .......... 11

- What is SOAP? When to use SOAP? ......................................................................... 11

- How can we rip out the business logic from PowerBuilder and make a SOAP service that can help our Migration Strategy? ................................................................. 11

LINK Code Conversion ................................................................................................. 12
Overview

Background

The Department of Children and Families (DCF) is replacing their current child welfare system, LINK, with CT-KIND (Kid’s Information Network Database) to comply with the Federal Comprehensive Child Welfare Information System (CCWIS) regulations, and promote efficiency and streamline processes in practice. This revamp is a fresh start as it is a complete redesign leveraging the newest technologies to meet the needs of the staff and the families served based on updated business workflows that have been through a LEAN process. The LEAN process ensures that the work performed is as efficient as possible, and the process is updated per any legislative or policy changes. By basing this rewrite effort on updated workflow processes, we are making sure we are cutting out steps that are part of the current process that do not bring business value.

In order to reduce the risk of developing such a large solution covering many areas of child welfare and comply with the Federal requirements, DCF is developing the new system in modules, one module at a time through Agile Development, using the SAFe methodology. This will allow DCF to deliver value to our customers quicker and incrementally, and to course correct if needed. It allows DCF to tackle the complexities of converting parts of LINK, minimizing the risk. By doing development modularly with the assistance of a vendor pool, it allows DCF to build in competition amongst vendors to deliver a quality product, and also allows DCF the flexibility to switch vendors if they are underperforming.

Because of the requirements mentioned above, and because of all the benefits of Agile Development, a natural question arises. How do we do development modularly? and, How do we ensure we develop each module individually and ensure the rest of LINK continues to function?

Strangler Pattern

What is the Strangler Pattern? The idea of the Strangler Pattern is to incrementally migrate the monolith by replacing pieces of functionality gradually until eventually, all of the monolith is replaced. As each functionality is replaced, the monolith is “strangled” allowing it to be decommissioned.

Goals

With the following strategy, DCF will be attempting to convert LINK from a set of older technologies to newer technologies. Because of the inherit risk of converting a large legacy system, we need to adhere to a set of values which will minimize the risk of the conversion. DCF’s existing legacy system, LINK, is comprised of the following technologies:
CT-KIND will be built on the following technologies:

- Microsoft Dynamics 365 (on premise)
- SQL Server/SSIS
- BizTalk Server 2016
- SharePoint 2016
- MDM VisionWare
- Microsoft Exchange Integration
- SSRS
- MicroFocus UFT
- Stress Stimulus
- PowerBI
- ID.ME
- RSA
- .NET/ Javascript/EXTJS

**Minimize Change to Monolith**

In many legacy strategies that are described in the industry, there is the reference to the legacy system as the monolith, a term believed to come from other industries and could have different meanings. However, for the sake of this strategy, we will refer to it interchangeably as the monolith or simply Legacy to describe all the systems that make up the legacy application.

Here are some of the disadvantages and advantages of a legacy system:

**Disadvantages:**

- Difficult to understand system functionalities
- Hard to find and separate business logic from User Interface and Data Access
- No documentation
- Difficult to integrate with current systems
- Hard to enhance or modify because unit tests, regressions tests are not available

**Advantages:** Able to support existing business processes.

Since the monolith is a very large, complex system with many different pieces to it, it will be very difficult, nearly impossible to make changes to its data model or database, or modify its processes. Therefore, one of DCF’s values would be to avoid the tendency to change or modify the existing database model.
The goal is to replace the monolith, not to keep adding to it. Every additional piece of code, even to support the conversion process, builds additional dependencies on legacy code. Therefore, in order to minimize risk, DCF needs to avoid the tendency to write new functionality as part of the monolith, unless absolutely necessary.

**Minimize Creating More Dependencies**

Where DCF needs to write code as part of the monolith, it should be done in such a way in which additional dependencies are not built on the monolith if possible. So, another one of our values would be to not build additional dependencies.

**Create Business Defined Strangler Scope**

Additional planning will be needed to determine how to apply the Strangler Pattern for DCF with our set of technologies and the requirement to keep the monolith running. In particular, the risk is high when the monolith is not built using an API, has no easy way to intercept database calls, or when the code is complex and the business logic, data access, and GUI layers are intermingled and intertwined. In order to minimize this risk, DCF needs to strangle a vertical slice across all layers at a time. The vertical slice should include everything required to perform the business workflow function end to end. The vertical slice is defined to include all the layers required to define a particular workflow scope:

- User Interface
- Business Rules
- Service
- Database Logic
- Security
- Reporting

Therefore, the value is to keep the scope of replacement at the “complete business workflow scope” keeping in mind the following:

- Minimize Legacy Database Model Changes
- Minimize Legacy Code Additions
- Single Business Process Scope

**Replacement Strategy**

Planning will be needed in order to determine the steps in the replacement strategy such as:

1. Impact Analyses for each process to identify the scope
2. Identify Data Conversion Scope
3. Identify User Impacts and security Permissions
4. Identity Methodology Technology or tools being used to implement the replacement functionality

Methodologies/Tools

In order to achieve our goal we have to rely on the strengths of a multitude of technologies. Often times there are multiple ways to achieve and deliver a business goal and value to the client. However each time we replace a piece of functionality with new methodologies and tools, we must make a best case effort to do cost-benefit analysis, and pick the Methodologies and Tools that have the best long term benefit and outcome. Some of the technologies part of our “tool belt” which make up what is known as the “Technology Stack” are listed below with insight how those tools can help achieve goals and solve problems.

Enterprise Service Bus (ESB)

What is ESB?
BizTalk ESB is an Enterprise Service Bus (ESB) used for message-based communication between disparate software systems for Enterprise Application Integration (EAI). EAI is based on a set of rules and principles for integrating application together over a bus-like infrastructure. Different applications can be integrated by putting a communication bus between them to enable the application to talk to the enterprise bus and vice versa. The ESB bus decouples the application from each other.

What problems does ESB solve? Why use ESB?
It is inevitable that LINK and CT-KIND need to communicate to achieve their functional goals during the strangler period. During the strangler period, both systems should make minimum, direct dependencies. BizTalk ESB abstracts direct dependencies between the systems by acting as a middle layer. Decoupling the dependencies is achieved by performing the data mapping, transformation and data delivery between LINK and CT-KIND. BizTalk ESB provides content-based delivery of messages based on the source data to the target system such as SQL or DB2. During the process of routing/delivery, source data undergoes transformation within the ESB mapping layer, as required for the target systems (SQL or DB2) based on business rules depicted in the ESB. ESB achieves routing of the source data to multiple target system based on target subscriptions to the intended source message data.

What is guaranteed message delivery? How does it work?
Messages and message subscriptions are stored in the BizTalk SQL database. The messages are persisted in DB until all the subscribers consume the source message. If the message is failed to deliver to the target system, BizTalk ESB retries the message delivery. If the target fails even after the retry, failed messages are routed and collected for error notification. Failed process/messages are hydrated in BizTalk SQL and can be resumed (mostly) from the point where it failed by using resume functionality associated with orchestration and message processing. This ensures guaranteed message deliver.
BizTalk ESB uses alternate delivery mechanisms if the delivery is failed with the primary target, as it will then send the data to the secondary using a backup transport. This ensures guaranteed message delivery.

How can the ESB be used to solve our conversion/migration needs?

- BizTalk ESB detecting data changes on SQL or DB2
- Support for SQL and DB2 out of the box adapters
- Support for security and reliable data delivery
- SSO based data delivery
- Data mapping based on static data mapping
- Dynamic data mapping based on the content of the data
- Transformation of the data using XSLT
- Delivering data to both SQL and DB2
- Distributed transaction between SQL and DB2 database
- Abstracting the workflow process for the EAI at the orchestration layer
- Dynamic routing of messages based on itinerary
- Handling the batch or real time processing
- Error and exception of failed data transfers
- Real time scaling and BizTalk throttling
- High availability and disaster recover

Extract Transform Load (ETL)

What is ETL?

Extract Transform Load (ETL) is a process for moving or copying data from one data source to another, such as copying existing data from the legacy LINK database into the CT-KIND database, or copying new/updated data from CT-KIND back to LINK. This process allows for data to be available for functionality that exists only in LINK (because it has not yet been implemented in CT-KIND). It is typically, though not exclusively, used in data warehouse applications, where data from multiple and disparate sources are brought together into a common structure to be analyzed, or processed together.

“ETL” stands for the three separate steps involved in the process: “extract”, “transform” and “load.” The extract step of the process is where selected data is pulled from the source database. There may be business rules established to define exactly which data is extracted from the source database, such as data that has been created and/or updated since the last synchronization of the target database.

The data is then manipulated, or transformed, via a set of business rules so that it will conform to the new target database design/structure. An example of a transformation might be a case where sex code (i.e.: male or female) is stored as “1” (male) or “2” (female) in the source database, but it is stored as “M” or “F” in the target database. The transformation step will convert all “1”s to “M”, and all “2”s to “F.”. Another kind of transformation might be a derivation in which data from the source database is calculated into a completely new field, which is then stored on the target database. An example of a derivation might be in a retail sales database which stores “quantity sold” and “unit price” for items, but
your target database only stores the “sale amount” for an item. The transformation would calculate “sale_amt = qty_sold * unit_price” and only store the calculated sale amount value in the target database.

Finally, the extracted and transformed data is then loaded into the target database. The structure of the target database may be very different from the source, so this would be handled by both the transform and load steps. For instance, data about people may be stored in one table in the source database, but it may be stored in three related tables in the target database. The transformation and load steps would split the data up appropriately so that it is stored in the target database in the correct tables with the correct relationships to connect the data together.

What are some ETL tools that we can use?
There are a number of different ETL tools available on the market, such as “SSIS” (which is integrated into Microsoft’s SQL Server database product), “Infosphere Datastage” from IBM, “Informatica” from Informatica Corp., “DT/Studio” from Embarcadero Technologies, and others. Since the design of the CT-KIND is Microsoft-centric and is targeting SQL Server as the database platform, SSIS would be the logical ETL tool to use for CT-KIND unless we encounter any technical issues or other reasons which would lead us to explore other ETL tools as a solution.

How do you identify the data to be converted and loaded into CT-KIND?
Coming up with an ETL strategy for moving data back and forth between LINK and CT-KIND will not be a trivial effort. First, the CT-KIND database will likely have a very different structure from the legacy LINK database, so the differences will need to be well defined and documented so that data can flow smoothly from LINK to the appropriate place(s) in CT-KIND, and vice-versa. The method for documenting this cross reference between the two databases will need to be determined, designed, implemented and maintained throughout the course of the CT-KIND Project. This cross reference will likely be a key player in the ETL process as it will help with the mapping of data back and forth in both the transform and load steps of ETL.

Just because it can be converted doesn’t necessarily mean it should?
Second, it is possible that all of the data that currently exists in LINK may not need to be brought over to CT-KIND. There may be some data elements that have traditionally been captured and stored in LINK that are not currently being used by the business. Perhaps it was for some piece of functionality that once existed in LINK that is no longer used, or perhaps it was captured “just in case” there would eventually be a need to use it in the future, and that need never materialized.

CT-KIND should be designed so all of the data that it needs to function resides in the CT-KIND database, and it should never have to go back to LINK to pull in additional data while processing. This is a very critical design consideration as the eventual end state of the CT-KIND Project will be one in which LINK no longer exists. The LEAN events that help define the detailed scope of each Statement of Work (SOW) for CT-KIND should help define exactly which data needs to be brought over to CT-KIND, and which data should not. At the end of the project, once the last SOW has been completed and CT-KIND is fully operational as a replacement for LINK, the only data left in LINK that has not been brought into CT-KIND should be the data that is no longer needed.
How do you handle the differences between the data models?
One final consideration regarding data: during the interim period between the time the first piece of CT-KIND becomes operational and the time the final piece becomes operational (and LINK therefore is no longer actively used), it will need to be established whether CT-KIND or LINK is the “book of record”, or “source of truth” for each piece of data that is common across both applications. Data should only be created or updated by the application in the “source of truth” database, and be read only for the application in the other. There should not be any situations in which a common piece of data is able to be created, updated or deleted in either/both databases. Whenever a common piece of data is created or updated in the “source of truth” database, that data will need to be replicated to the other database such that both databases stay in sync as closely as possible. This is where the ETL processing that governs the flow of data back and forth between LINK and CT-KIND will play its most critical role.

Robotic Process Automation (RPA)

What is RPA?
Robotic Process Automation (RPA) is a software application that can be configured to perform interactions with other existing applications in a human-like manner (using keyboard mouse inputs).

How can this technology help us?
The RPA technology can help by improving productivity by allowing users to see one single unified UI and interact with 2 or more systems seamlessly (for example data entered in the RPA UI can be asymmetrically split or duplicated and entered into both CT-KIND and LINK simultaneously as configured). With the Strangler Pattern, RPA can be used to maintain new system information up to date into the old system while the old system still needs to be updated and functional (until decommissioning). Search and prefill working across secondary systems can speed up work in the primary system. RPA can encapsulate and simplify working with a certain system by exposing selective functionality to another system. Once the system is encapsulated for a certain workflow point of view, the staff can now ignore the encapsulated system for that particular workflow.

Security

What is the role of security in the migration strategy?
Every piece of functionality needs to be driven by Permissions and Roles so that proper access to confidential areas is maintained. Whenever functionality is being called whether through the UI or through the API security and permissions must always be enforced. Please see table below:
### Permissions

<table>
<thead>
<tr>
<th>What is it?</th>
<th>A permission is a string describing a system function that the system conditionally restricts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>How does it work?</td>
<td>If an entity attempts to access the system to perform a certain function, the system will only allow access if entity already holds the permission matching the requested system function.</td>
</tr>
<tr>
<td>Analogy</td>
<td>Permissions are like individual keys to different functions of the system.</td>
</tr>
<tr>
<td>When to use</td>
<td>Permissions are used in code to control security at the system function, detailed level.</td>
</tr>
</tbody>
</table>

### Roles

<table>
<thead>
<tr>
<th>What is it?</th>
<th>Roles are sets of permissions under one name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>How does it work?</td>
<td>A unique type of system accessing entity would be assigned a unique role for accessing the system. That role controls what permissions that entity holds which in turn controls what system functions the entity can access within the system. This way different types of entities are given different access to the system according to prescribed business needs.</td>
</tr>
<tr>
<td>Analogy</td>
<td>Like key rings, these roles group different sets of permissions (keys) under one name (key ring).</td>
</tr>
<tr>
<td>When to use</td>
<td>Roles are used in security administration (not in code) to allocate complex security schemas to different system accessing entities, at the entity level (higher level grouping of permissions). This controls what functions a certain type of entity can access within the system.</td>
</tr>
</tbody>
</table>
PowerBuilder Web Services

What are web services? REST vs SOAP? What are PowerBuilder specific web services?
Web services are a standardized way of integrating applications using industry standard common protocols such as XML, SOAP, REST, JSON, and UDDI over an Internet protocol. In simplest terms, it is a method of communication between two applications or electronic devices over the internet. They can be used to exchange information or perform a service. SOAP was the initial standard protocol, followed by REST and JSON. There are no PowerBuilder specific web services, but SOAP web services can be created and consumed from PowerBuilder. PowerBuilder web services are deployed as .NET web services in an IIS environment.

REST stands for Representational State Transfer. REST is a lighter weight alternative. Some find it easier because you don’t have to use strictly formatted XML, instead using a formatted URL. REST uses 4 HTTP commands – GET, POST, READ, and DELETE to accomplish CRUD (Create, Read, Update, and Delete) tasks. REST can also output in different formats in addition to XML – CSV (Comma Separated Values), JSON (JavaScript Object Notation and RSS (Really Simple Syndication).

Some prefer SOAP because of its strict guaranteed format of request and response, and some dislike it for those same reasons, but both are effective.

What is SOAP? When to use SOAP?
SOAP stands for Simple Object Access Protocol. It is a standard that was created and documented to meet the necessity of disparate applications, possibly using different technologies to communicate with one another over the internet. The language/format used is XML, which stands for Extensible Markup Language, and has become the industry standard for communicating data worldwide. SOAP is the easiest protocol to use and offers very well defined requests and responses. PowerBuilder can only work directly with SOAP web services. REST services can be called from PowerBuilder, but need a “wrapper” or transformation to SOAP.

How can we rip out the business logic from PowerBuilder and make a SOAP service that can help our Migration Strategy?
Before doing this exercise, the requirements of the service should be well defined. Each business function should be broken down into the services that satisfy that function. Then the common services need to be evaluated across each function they are used to make sure they are being implemented the same. Once this is finalized, the PowerBuilder code needs to be interrogated to pull out all of the business rules. Business rules/edits/validations can be located in data window controls and functions inside windows or user objects. Once all of these have been identified, a service can be created that satisfies all of the conditions while performing the service.
LINK Code Conversion

Sections
PowerBuilder is groups of objects:

a) Applications  
b) Data Windows  
c) Functions  
d) Menus  
e) Structures  
f) Users  
g) Windows

These objects reside in libraries called PBL grouped by unit of work, for example CPS is in SM01a.pbl and investigation is SM06a.pbl. LINK is made up of 75 PBLs. There is a process to compile all the PBLs into an executable object and all the PBLs are compiled into PBD. This is what gets deployed.

All objects have events that code can be used to do processing. Everything a user does on a window can be captured in an event. All objects also have functions that can be called in the script to execute code and can be called within the object. There are also global standalone function objects that can be called from anywhere within the application. This could be formatting a name when all you have is the ID. The top level object of LINK is the application. The application is made up of many windows that divides the unit of work. Most of the windows are inherited with business logic included. Each window has business logic unique to that window. On some of the windows, there are multiply tabs that separate subjects are addressed. On the window are user objects that are also inherited that has business logic, like date column edits. On some user objects, there are data windows. The data window is the communication between the screen and the database. Within the data window is more Business Logic that includes validation and allowing, or not allowing, access to a column.

Batch Process

CT-KIND functionalities in the first SOW encompass Case Maintenance and Person Management functionalities and will require the decommissioning of related batch processes in LINK. The following observations underline some ideas that should be taken into consideration in the decommissioning efforts.

The new functionalities for Case Maintenance and Person Management developed in CT-KIND will be distributed only to Careline workers in the first SOW. As a result of the initial targeted audience, the associated batch processes that support these functionalities in LINK will need to remain active as workers outside of Careline will not have any accessibility to these new functionalities.

During this phase of the implementation, any updates performed in CT-KIND or LINK will require data synchronization to be reciprocated bi-directionally. For instance, if a worker requests for a Case Merge/Person Merge in CT-KIND or LINK, the information that is being copied over from the old
case/PID to the retained case/PID will also need to be captured and synchronized in both CT-KIND and LINK in order to maintain data integrity.

There are several options that can be taken into consideration for the data synchronization. One method is to create DB2 triggers to indicate that an event has occurred. This event trigger can be housed in a staging repository and can be interrogated intermittently to check for new requests/transactions of updates to CT-KIND or LINK. The other alternative is to utilize ESB in BizTalk. This Enterprise Service Bus serves as the middle layer to manage any data mapping/conversions utilized in the synchronization process between LINK and CT-KIND.

In order to maintain this data integrity, all of the entities identified, such as DB2 tables, having any association with the Case Maintenance and Person Management batch processes will need to remain active due to dependencies that exist beyond the Careline Module.

Another important aspect of the decommissioning of LINK batch is job dependencies. Job dependencies need to be identified for impact analysis. Predecessors, successors and interfaces from an external agency or vendor play a pertinent role. Any unplanned disconnect to these job dependencies can negatively affect the current business rule.

From the perspective of the first SOW, jobs that pertain to the Case Closure, Case Merge, Person Merge, Person Delete and Expungement processes have dependencies. Case Closure precedes Case Merge and Person Merge precedes Person Delete.

Additionally, impact analysis on ESP job scheduling needs to be done to ensure that once a job is deactivated, it does not interrupt the remaining job stream. Coordination with DAS/BEST is needed as they are the gatekeeper of the ESP Job Scheduler.

**Master Data Management (MDM)**

The functional capabilities of a successful integrated and configured Master Data Management (MDM) solution will ensure accuracy, quality, consistency and availability of our data so that it can be shared across the agency, as well as within different business processes. It will enable DCF to break down the inefficiency of data silos by creating and maintaining a single and complete view of the children and families across the board. Enabling this “single view of the truth” is an essential pre-requisite to leverage the benefits of a data driven agency such as DCF, including improved customer engagement of children and families and operational efficiency alongside reduced regulatory risk, resulting in significant improvement in accurate reporting and fact-based decision-making.

The key capabilities DCF is looking to achieve are as follows:

Master Data Services is a master data management platform that will enable DCF to create and maintain a single, authoritative standard for business data leading to fewer errors and less redundancy in business processes, and streamlines them to provide a more efficient and effective system of delivery based on this trusted data.
**Key features include:** Data Cleansing, Data Matching, Duplicate Detection, Data Merge, Create and Maintain the Golden Record, Accurate Reporting, Comprehensive Data Stewardship of the citizen records (child, parent/adult, etc.).

Verifying data improves data quality through reference data lookup facilities enabling search features to be more effective and accurate. MDM ensures that data is kept current, accurate, complete and relevant.

**Key Features include:** Synchronizing data between MDM and LINK (internal sync), Data Enrichment (enhance, refine or otherwise improve raw data, e.g., correct likely misspellings or typographical errors in a database through the use of precision algorithms).

Governance processes have been implemented to reflect the business and improve sharing and synchronizing of external data, supporting the decision-making process and distributing the most consistent, accurate data.

**Key Features include:** Data Synchronization with external systems (EMPI), Data Consistency, Enhanced Governance and Change Verification, as well as Maintaining Data Integrity, Data Versioning and Auditing Integration with Dynamics will provide accurate, current and consistent data, which will improve service delivery and enhance the user experience.

**Key Features include:** MDM within CRM (Microsoft dynamics), Unified User Experience.

A Master Data Security strategy must be in place for both data in motion and data at rest. When there is no MDM solution in place, the risk profile for the organization is higher because sensitive data is distributed across systems. The organizational exposure to certain types of business risk, such as critical information loss and access to unauthorized information, can rapidly spread out of proportion.