CSC492 Final Report: Cormant CS-EDIS

Jarred Stelfox
# Table of Contents

1 Introduction .................................................................................................................. 1
  1.1 Description of the Problem ...................................................................................... 1
  1.2 Overview of the Solution ......................................................................................... 3
  1.3 Scope ....................................................................................................................... 4
  1.4 Objectives ............................................................................................................... 5
  1.5 Outline of the Report ............................................................................................. 6

2 Scenario of System Use ............................................................................................... 7

3 EDIS at a High Level ..................................................................................................... 8

4 EDIS Layers in Depth .................................................................................................. 10
  4.1 The External Extraction Layer ............................................................................... 10
  4.2 The Mapping Layer ............................................................................................... 10
  4.3 The Syncing Logic Layer ...................................................................................... 11
  4.4 The Internal Extraction Layer .............................................................................. 11

5 Testing and Validation ................................................................................................. 13
  5.1 Creating a Component in an External System ....................................................... 13
    Figure 5-1 – Creating a new Server in Salesforce ....................................................... 13
    Figure 5-2 – Save the component ............................................................................ 14
  5.2 Sync Component to Cormant-CS ......................................................................... 15
    Figure 5-3 – Open Cormant-CS .............................................................................. 15
    Figure 5-4 – Validate data sync .............................................................................. 16
  5.3 Moving a Component / Changing Attributes ....................................................... 17
    Figure 5-5 – Moving Server .................................................................................... 17
    Figure 5-6 – Updating an Attribute ........................................................................ 18
    Figure 5-7 – Verifying Data Sync Back to Salesforce ............................................. 19
  5.4 Create an External Order ....................................................................................... 20
    Figure 5-8 – Creating an Order in Salesforce .......................................................... 20
    Figure 5-9 – Order Detail Page ............................................................................... 20
  5.5 Sync Order to Cormant-CS .................................................................................. 21
    Figure 5-10 – Verify Order Synced to CS ................................................................. 21
    Figure 5-11 – Verify Order Synced to CS In Detail .................................................. 22
  5.6 Add Task to Order / Sync Back to External System ........................................... 23
    Figure 5-12 – Create a Order task in Cormant-CS .................................................... 23
    Figure 5-13 – Task Details Page ............................................................................. 24
    Figure 5-14 – Task Details Page In Detail ............................................................... 25
    Figure 5-15 – Verifying the Order Now has the Associated Task ........................... 25
    Figure 5-16 – Verifying the Order Now has the Associated Task ........................... 26

6 Related Work ............................................................................................................... 27

7 Conclusion ................................................................................................................... 28

8 Future Work ............................................................................................................... 29
  8.1 Efficiency ............................................................................................................... 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.2</td>
<td>Configuration HTML Front-End</td>
<td>29</td>
</tr>
<tr>
<td>8.3</td>
<td>Event Driven Syncing</td>
<td>29</td>
</tr>
<tr>
<td>8.4</td>
<td>Configurable Syncing Directions</td>
<td>30</td>
</tr>
<tr>
<td>8.5</td>
<td>API for EDIS</td>
<td>30</td>
</tr>
</tbody>
</table>
1 Introduction

Data is paramount when it comes to the Data Center Infrastructure Management (DCIM) industry. Some organizations are using as many as 40 different systems to manage all the data within their ecosystem. Whether you are using tools for tracking a server's configuration in a Configuration Management Data Base (CMDB), linking a customer to a piece of equipment in a Customer Relationship Management tool (CRM), or tracking an issue in an Information Technology Service Management tool (ITSM), data is the vital component. Often a full ecosystem is a conglomerate of these management tools, but data from one system is often most helpful when married to data from another system. Integration of data is a growing pillar of the DCIM market, as it saves the end user's time, resources, and aids with automation, by providing disparate data formats in a single, efficient interface.

The industry lexicon for an integration is a “connector,” which implies a short, simple, and easy to start integration. This market opportunity is driving to create an integration system. The goal of this integration project is to answer the market demand: making assimilation of data across any type of management system easy, quick, and seamless to the end-user.

1.1 Description of the Problem

The company I work for, Cormant Inc., is a Data Center Infrastructure Management (DCIM) company based out of San Luis Obispo, CA. Cormant's cornerstone product is called Cormant-CS. This product has many features, all of which were created with a common purpose: to help asset tracking in a data center.

At the core of the Cormant-CS software is physical asset tracking, cable connection tracking, and a workflow to assure connections or equipment are installed in a logical fashion. On top of this, Cormant-CS tracks information about individual pieces of equipment, both
physical or virtual. The equipment data, in terms of our software, are called attributes.

As the DCIM industry is a small and growing niche market, customers are the force driving market changes. Over the last few years, the market has shifted and users have placed value on three critical criteria:

- Scalability
- Investment in predictive analytics and automation technologies
- Ability to integrate with other data center tools

The inception for Cormant-CS was to have it be extremely scalable and allow automation from internal C# scripts. This accounted for two of the three critical criteria. The third critical criterion was added in release version 8, and we now have predictive analytic components. However, even with meeting these three critical criteria, our integration tools have been inadequate for a fast-turn-around.

Customers are expecting “connectors” or easy integrations to their internal programs. They don’t want an integration to be a long, costly process. They are expecting that we have incorporated tools or features to very quickly integrate with their systems and for these integrations to be straightforward and easy to write.

Any given data center has a different client base and use case. This means that one data center may rent their machines out as a cloud service, while another data center may track computer terminals, and their connections, in an airport. Examples can be found by looking at Salesforce and ServiceNow, which are the two most talked about integrations. The most common example of a Salesforce integration is for a cloud service that needs to update information about which computers are owned by which customers. Where as, ServiceNow is an IT Service Management solution that people often use as their internal ticket system.
It is the fact that the Cormant External Data Synchronization Service is able to synchronize data from any management tool across Cormant-CS, which makes the integration system an engaging project. The project should be able to dynamically update assets, tickets, customer data, power consumption and project data in Cormant-CS or an external system. In addition, the project needs to be able to consume and export data of varying types while treating all implementations as equals. Equals means that all Application Programming Interfaces (APIs) are abstracted to use the same programming interface.

1.2 Overview of the Solution

The simple idea of taking an object of any type, which has associated values, in one system and syncing it to data in Cormant-CS is the birth of the Cormant External Data Integration Service, or EDIS.

At a high level, the external integration system needs to be able to extract any kind of object from an external system and synchronize information to an object within Cormant-CS. Conversely, data will need to be pulled out of Cormant-CS and pushed into the external system. This means that data will need to be extracted and translated into a format that is able to synchronize to Cormant-CS.

For the needs of the market, this solution should run automatically. The best way to do this is to have it be a service, which starts as the machine boots. Since Cormant-CS already uses a polling method for most actions, having the service configurable to run every every ‘n’ number of minutes is sufficient, where ‘n’ is typically every 15 minutes.

To reduce coupling to any API, EDIS will encapsulate the implementation details of each integrated tool. This imposes a need to implement a required interface. This will keep all the core syncing logic loosely coupled to any particular service.

Part of the core functionality of syncing data is mapping values from one system to another. This mapping allows for data to transfer
fluentlly. Thus, EDIS must be able to map details between an object in each system. Since each type of tool handles things a bit differently, there is need for the concept of sync type. Specific to Cormant-CS, which is the main focus of this project, sync type will utilize Cormant-CS terminology, such as an asset, attributes of an asset, workflow orders, and workflow tasks.

All of these features should culminate into EDIS being a "connector" service. This means that all of the core syncing logic will be handled within EDIS.

A "sync-able" object will be pulled into the service when

- This object is compared to the Cormant-CS object via a primary key between the mappings
- The sync-able object has it's mapping updated to match the most recent data
- The object is then synchronized to the appropriate Application Program Interface (API) – either Cormant-CS or an external service.

Looking at the big picture shows why it will be a "connector" service. For each integration, it will require wrapping the external service's API and extracting values into and out of a "sync-able object." What this really means is adding more services should be a matter of wrapping the API and object conversion. This over time will allow for integrations between other products to be easy and quick.

1.3 Scope

As a company, we have many customers interested in integrations, but have not received requirement statements from them. Thus, the general scope of EDIS has been determined from discussions among many vested parties including the project managers, sales, customer relations, support and the development team. We have also researched needed features from the current market demands.
The scope defined for the integration framework is massive and will be a continuing project after my graduation. An overview of the total project scope is listed below:

- Service to run every ‘n’ number of minutes
- Authenticate to external services
- Wrap all APIs and use interfaces to decouple core business logic
- Parse external data and pass data into the core logic
- Map values of one system into another
- Have configurable syncing directions
- Have a test mode
- Have HTML pages, which allow for configuration of mappings as well as any system level settings such as the run time
- Users will be notified and asked for an action when there is a data issue or data conflict

Parts of the EDIS project are expected to take a few months and rely on other internal features being written and implemented into Cormant-CS. Due to this, the scope for this paper and my prototype is based around core syncing logic. The true goal is to have a prototype, which can be a basis of growth and features can be added.

Requirements for the prototype are:

- Service to run every ‘n’ number of minutes
- Authenticate to external services
- Wrap all APIs and use interfaces to decouple core business logic
- Parse external data and pass data into the core logic
- Map values of one system into another

1.4 Objectives

The objective of this project is to learn more about our core product, while helping meet the current market demands. In order to do that, this paper is not just single use. It is written in a way, that should help another developer understand the structure behind the implementation, without releasing information about the source.
1.5 Outline of the Report

The following sections of the report detail a more technical view of the External Data Integration Service. This includes a flow of interaction on how to use the solution developed via this project. There is also a discussion of design details and motivation for these details. In addition, Section 5 discusses validation of the data synchronization. Finally, Section 7 is the conclusion, which also discusses future efforts.
2 Scenario of System Use

As this project is being done “in-house” and without access to a live customer management tool, the External Data Integration Service needs to use test data sets. For this purpose, it used a real management tool, which is in the cloud, to pull simulated data out of that tool.

We had the ability to use one of our Salesforce Sandboxes to stand up the data for testing this integration. Salesforce is traditionally thought of as a Customer Relationship Manager and a sandbox is the term for an empty database, which can be reset with the push of a button. Other than that difference, a Salesforce sandbox is the same as a traditional Salesforce account.

For those who are not familiar with many management systems, they often allow you to configure a database to fit your data. They are quite flexible. Since this test bed used Salesforce, I created some tables in Salesforce to hold the data, which will be synced via EDIS.

Once that was done, I had to enable an API endpoint to be able to programmatically access the created tables. Salesforce allows you to write your own methods, which will be accessible from the API endpoint. Although this means you generally can write some very interesting methods, the only needed methods are basic getters and setters for each table. In short this means that as long as an API supports getting or setting an object, EDIS will too.
3 EDIS at a High Level

Figure 3-1 Data Process –High Level

Figure 3-1 provides a high-level illustration of the data process. It shows how data will flow through the system.

After some discussion within the development team, it seems best to split the framework into differing layers to help encapsulation. Each layer will abstract the complex implementation details to the following layer. As is expected of any good software, each layer encapsulates all functionality within its own scope. The Cormant-CS EDIS layers are as follows:

- The External Extraction Layer
- The Mapping Layer
- The Syncing Logic Layer
- The Internal Extraction Layer

The Extraction layer takes data out of the external tools. This layer’s purpose is easy to understand. It simply extracts data.

The CS EDIS Mapping layer is the true entry point of data. Having an API will allow for the external system’s API and implementations of extraction to vary at the extraction layer.
The CS EDIS API will use a ubiquitous proprietary format for all objects, such as dates, throughout the rest of the framework.

Once data has made its way into CS EDIS, the Cormant-CS API will be used to find the corresponding (or new) objects and determine which way a sync will need to happen. This is the Syncing Logic Layer and is where hardcoded business rules will be applied if they are needed. The integration service will read from configuration files and map values between each system.

The final layer is the Internal Extraction Layer. The Cormant External Data Integration Service will be easier to understand if you think of Cormant-CS as a “sync-able tool.” This means that every external tool and Cormant-CS will actually be treated only as a “sync-able tool.” Thus, each tool will have the same interface methods. This decouples any need to manage specific implementations from either the internal or external APIs when extracting data. Thus, allowing for a more agnostic and generic solution.

The data integration service will include logging, web screens, configuration, conflict resolution and more. Data will sync to Cormant-CS and to the external system according to the configuration.
4 EDIS Layers in Depth

4.1 The External Extraction Layer

The external extraction layer will oversee authentication, extraction and conversion of data from an external API. All authentication information will be read from a configuration file. This will save the external data integration service from needing a new compilation on each password change.

Each external service will have some API or way to extract data. The conventions of each API, and the underlying objects, will likely be different. To ensure that Cormant-CS EDIS is able to easily connect with any system, these API’s will be encapsulated. Encapsulation will ensure the external system’s representations of objects are hidden from the rest of the syncing code. Thus, the external extraction layer will parse the underlying objects of the external API and pass them on as a “sync-able” object. This layer will also be responsible for parsing a sync-able object back into the representation of the external system and send that object back to the external API.

4.2 The Mapping Layer

Along with this layer comes mapping attributes of one API object type to attributes of another API object type. Since the internal and external representation of objects will be converted into the same “sync-able” form, mapping from an attribute name in one system will be correlated and mapped to an attribute name in another system. An example would be an attribute such as a ticket number field name being "ticket number" in the external system, which is mapped to "Ticket no." on the internal representation. This will allow for configurable screen, which transfers attribute names from an external system to the ones in our internal Cormant-CS.
4.3 The Syncing Logic Layer

The next layer is paramount to correlating the disparate data. Now that we have objects from the internal system and objects from the external system that have the same mapping of attributes. In doing this, we need to determine which data to sync and which direction to sync them. Not only will this be a configurable option, but a two-way sync will actually need to know which direction an object needs to go to be synchronized correctly. Any kind of changes within the business rules will happen here. It is the main center of logic controlling the flow of data. Any optimizations to the business rules would happen here. For example, it would be possible to check if anything has changed from the internal or external system. If nothing has changed, then no action needs to be taken. These kind of optimizations will be added much later once the framework has reached a functioning phase.

4.4 The Internal Extraction Layer

There should be a general note here that the internal Cormant-CS API will also be treated just like any other API. The overall goal is to use the same known form for internal Cormant-CS objects and external objects. This guarantees that the syncing logic and business rules do not need to worry about the implementations of either the internal or external side. Thus, the same idea applies: Wrap the API and conversion to and from a “sync-able” object.

All syncing logic and determination of mapping will be easier due to transferring data between the same type.

There will, however, be a notion of a sync type attached to the “sync-able” object. This is due to the limitations of having a configurable syncing service. As previously mentioned, this service can sync everything from a task of plugging in a cable to a server to updating the customer name on a piece of hardware. This configurability requires a knowledge of what type of data to get from each system.
This will eventually be configurable, but that is out of the current project scope.
5  Testing and Validation

5.1 Creating a Component in an External System

A user will need to log into the Salesforce sandbox account, then click on the components tab. The component table has some values, which are needed and some that are optional. For the example in Figure 5-1, the “Serial #” field is used as a primary key in Cormant-CS to see if a component exists. The component-name will be the name of the component in both systems.

![Component Edit]

**Figure 5-1 – Creating a new Server in Salesforce**

Notice that the component is being created in Salesforce. This will cause the component to be synced across to Cormant-CS, but what happens if this component does not exist? The simple answer until conflict handling HTML screens are implemented, the component is moved under its site location to a “store room.”

Hence, this is why the location field was left blank. Once the data is synced across, a component will be added to Cormant-CS and the path location of that device will be synced back to Salesforce.
After clicking save, the user will see the component detail screen: Figure 5-2. It is important to note that location was left blank and the site is filled in.

**Figure 5-2 – Save the component**
5.2 Sync Component to Cormant-CS

Figure 5-3 shows that under the 02-ATL-DC1 site is a store room that is empty. It also shows that the current service is set to sync every 2 minutes.

![Figure 5-3 – Open Cormant-CS](image-url)
As shown in Figure 5-4, after two minutes, the data from Salesforce is moved across to Cormant-CS. Under the store room, a new component was created with the name DC1-01A-SVR08. The server is a Dell PowerEdge R720, which matches the Salesforce system. It can also be seen that the serial number matches across systems. This means any changes in Cormant-CS will sync back to Salesforce.
5.3 Moving a Component / Changing Attributes

Figure 5-5 is an example showing a user may change the server’s location by moving it into the first rack in the first row of this data center.

Figure 5-5 – Moving Server
As Figure 5-6 shows, a user may also change the server’s status within their data center.

**Figure 5-6 – Updating an Attribute**
Figure 5-7 is a screen-shot showing verifying data sync back to Salesforce. After two minutes, this data is synced back to Salesforce. The location is updated to match the path of the component. For simplicity, we show this path as a file path like string of text. The server’s status is also updated to primary, which matches the changes in Cormant-CS.

Figure 5-7 – Verifying Data Sync Back to Salesforce
5.4 Create an External Order

Figure 5-8 shows that the External Data Integration Service was created with the intention to sync many types of data including with the Cormant-CS workflow. Here you can see a user creating a new “Order” for three new servers.

![Figure 5-8 – Creating an Order in Salesforce](image)

Figure 5-9 is a detail page of an order is a bit uneventful currently. You can see that there is a list of “Workflow Tasks.” This is the Cormant-CS terminology for a task.

![Figure 5-9 – Order Detail Page](image)
5.5 Sync Order to Cormant-CS

In Figure 5-10, the order can be found in the Cormant-CS Workflow module. The user will notice the project code matches and the customer details match across the Systems. A user should also notice the Salesforce field “Account” mapped into the Cormant-CS attribute “Customer.”

Figure 5-10 – Verify Order Synced to CS
In Figure 5-11, the order is more viewable. The name matches the order created in the external system. In Cormant-CS this has “Order ID” 000004.

**Figure 5-11 – Verify Order Synced to CS In Detail**
5.6 Add Task to Order / Sync Back to External System

Figure 5-12 shows how a user now may want to create an install task for each server. For this example, the user only creates one of them.

![Figure 5-12 – Create a Order task in Cormant-CS](image-url)
Figure 5-13 shows how a user may want to view more information about the task in Cormant-CS. This will help verify the data moved across systems correctly.

![Figure 5-13 – Task Details Page](image-url)
Figure 5-14 shows the important syncing details. This shows it is an install task, which has a status of being planned. It is the first task of the “Order”.

![Task Details Page In Detail](image)

**Figure 5-14 – Task Details Page In Detail**

Figure 5-15 shows how user can view the order in Salesforce and verify the task was added to the order by looking at the “Workflow Tasks” section.

![Verifying the Order Now has the Associated Task](image)

**Figure 5-15 – Verifying the Order Now has the Associated Task**
If a user clicks on the task, Figure 5-16, they can see information about the task. This includes, the status of each task.

Figure 5-16 – Verifying the Order Now has the Associated Task
6 Related Work

Cormant-CS EDIS was based off another Cormant project, CS WIT. CS WIT was a syncing service created many years ago, in fact WIT stands for Web Interface Technologies. This project was used as a model for many core code abstractions. Thus, a lot of features and requirements for EDIS were built from the CS WIT. Some of these requirements are: directional syncing, generalized syncing of any type of data, and configuration mappings.
7 Conclusion

The development process of this project was a wonderful learning experience. It is the first project I have created, that will actually go into production and have a real-life-cycle.

I gained invaluable experience and learned many things while working on this project. Many of these things will be useful for the rest of my life. Examples are the features of the Visual Studio IDE, the design process, design patterns and even simple priority management.

The Cormant-CS EDIS project is vast and will continue to be created and upgraded for some time to come. There were many challenges in figuring out each external tool’s API. It showed me just how important documentation is. Moreover, a single example can go a long way to help a developer’s understanding.

The future of EDIS will likely rest in more features than core functionality. The most difficult challenges ahead will be dealing with conflict resolution, since this feature does not rely on any current logic. Another large challenge will be on the side of Cormant-CS: changing the API. After some work on this project, there are improvements to the Cormant-CS API that can make our API and EDIS more efficient and robust.

Overall, the creation of EDIS was a great introduction to what the future holds. The project was a powerful learning experience, which demonstrated the knowledge gained over the course of earning a Software Engineering degree from Cal Poly.
8 Future Work

As this project is going to evolve after my graduation, there are a few things that can be added or optimized.

8.1 Efficiency

The largest worry of efficiency for this system is extraneous queries to the Cormant-CS database. Some customers put the database on a wildly under-powered computer. Hitting the database too hard will utilize a lot of RAM on the machine running Cormant-CS and could impact its performance. The ideal solution is to determine if an object needs to be synced from Cormant-CS to the external system. This will make sure any objects from the database are loaded only when needed.

8.2 Configuration HTML Front-End

The configuration should mature into a nice HTML front-end to the service. This front-end should allow configuration of many system features. At a minimum, there should be syncing time, the mapping between systems, the type of each sync, and the direction of each sync.

8.3 Event Driven Syncing

Cormant-CS has a scheduler built in, which inherently uses polling. This is the core part of most time-based activities. Since there is no concept of an event based activity in Cormant-CS, it is not possible to trigger an update to an external system only when a certain object has changed in Cormant-CS. For example, an update of a component automatically pushing changes to an external service. This would make for a nice feature overall, but requires changing the core code of Cormant-CS.
8.4 Configurable Syncing Directions

Among some of the discussed business rules, many customers wanted to have one direction syncing of data. This would make one system the “master” of data with the other system just updating to match. A great use case for this is to synchronize data once. This is in effect transferring data from one system to another. From then on, the second system (the one to which data was sent) will now be the “master” of that data.

8.5 API for EDIS

A great way to expand on this project is to add an API. This would allow the external extraction layers to then connect to the API. Each external system would then talk to the API. In totality, this would decouple the external layers into their own service. The further decoupling of the external systems would mean adding a new external system. This is a matter of writing a new service and not a matter of updating the EDIS service itself.