codit

White Paper

Dynamics 365 Integration

Unlocking the Value of Dynamics 365 Through Integration

Author: Pim Simons | Integration Domain Lead and Senior Consultant

Introduction: Charles Storm | Lead Architect
Jonathan Gurevich | Business Development Manager

Reviewers: I would like to thank Sam Vanhoutte, CTO at Codit, Steef-Jan Wiggers, Azure Domain Lead and Senior Consultant at Codit Netherlands and Sagar Sharma, Consultant at Codit Malta for reviewing this white paper.

Version 1.0

Table Of Contents

1. Introduction: Architecture First! 4
1.1. Scope 6
1.2. Intended Audience 6
1.3. Future Versions 6

2. Defining our Approach 7

3. Integration Patterns 8
3.1. Overview 8
3.2. Providing Data 8
3.3. Retrieving Data 9
3.4. Receiving Data 9

4. Integration Capabilities of Dynamics 365 10
4.1. Overview 10
4.2. Web API 10
4.3. Custom Service 10
4.4. Event Pipeline 12
4.5. Common Data Service & Data Integration 12

5. The Integration Platforms 13
5.1. Overview 13
5.2. BizTalk Server 13
5.3. Azure Service Bus & Logic Apps 13
5.4. Microsoft Flow 15

6. Hybrid Implementations 16
6.1. Combining Azure Service Bus and BizTalk Server 16
6.2. Combining Azure Service Bus, Logic Apps and BizTalk Server 16
6.3. Including API Management 17

7. Choosing an Integration Platform and Method 18
7.1. The Platform 18
7.2. The Method 19

8. Conclusion 21

9. References 22

1. Introduction: Architecture First!

With Dynamics 365, Microsoft has entered a new era in offering ERP and CRM as SaaS solutions. From both a solution and technology point of view it is easy to recognize the tremendous value, scalability and other benefits that this platform offers.

Typically, applications like ERP and CRM are not the only applications in an organization's application landscape. It is common to have business processes transcend these types of applications, which requires a very critical accompanying aspect of every Dynamics 365 implementation: Integration. Having engaged with multiple customers and Dynamics implementation partners on this topic, we can confidently state that integration really unlocks the value of Dynamics 365 for end users, business partners and business processes. However, there does not yet seem to be mutual agreement and clarity on how to approach this key aspect.

Taking a step back to consider some of the lessons we've learned from the conversations we've had about these topics, it is clear that one truth stands out above all other lessons: a transition to Dynamics 365 is not just a transition of core systems like ERP and CRM to the cloud, it is a declaration for a transition to the cloud that surpasses ERP and CRM, and extends to a much broader scope, perhaps even the entire scope of IT. For all intents and purposes, one should assume that a choice for Dynamics 365 puts an organization on a track to a full transition to the cloud, with a hybrid transition period of varying length. A bold statement! But from an architecture point of view, something that needs to be planned for accordingly, to avoid having integration turn into a burden that holds this transition back, instead of being the flexible layer that enables this transition. So what does this translate to when considering the scope of integration?

As not all applications will immediately be replaced by cloud alternatives, an implementation of Dynamics 365 therefore launches customers into a hybrid IT landscape. Such a hybrid scenario requires an integration strategy that allows for seamless use of both on-premise and cloud assets and, perhaps more importantly, it must be fully prepared to allow for a further transition towards a (full) cloud based IT landscape.

This means that, when entering a Dynamics 365 engagement with your implementation partner or customer, the topic of integration should be covered at the onset of the Dynamics 365 project. Asking yourself the question of how to deal with existing and future integrations, for each separate integration requirement that will arise further down the line, is a luxury organizations can not afford. The Integration strategy, that should consist of an integration architecture combined with clear and standardized implementation guidelines, should be viewed as a critical success factor in both the short term (the initial Dynamics 365 implementation itself) and the long run (in which we continue down the path of further transition to the cloud).

This white paper contains lessons we've learned from our experience with over a decade of architecting, developing and maintaining integration solutions surrounding Dynamics ERP implementations, to which we've added the specific knowledge we've gained in multiple Dynamics 365 projects that we've already been involved in. This white paper focuses mostly on the implementation aspect of integration with Dynamics 365, and the question of which technology components to use. An important note however is that this is no replacement for architecture! To be more concise: this white paper assumes that an

integration architecture is already in place. This integration architecture should at least address topics of responsibility of components from an architecture perspective, as well as reference solutions for common integration patterns. The main reason why this is important is to avoid a technical driven integration solution, which has no foundation in architecture, and will always result in nothing but a point-to-point integration solution using fancy technology like BizTalk, Logic Apps or something else. As with any point-to-point implementation, this will result in an unstable and hard to maintain integration solution on both on a technical and operational level. Using advanced integration technology to build an integration solution like an ESB, without having architecture in place, is something we've more than once jokingly referred to as an *Enterprise Spaghetti Bus*.

Having a clear view on the target architecture for the coming years is key in determining how you would set up your integration landscape: Will you maintain periodic batch processing? Move to full API? Or are you aiming for an Event Driven architecture or even a Micro Services based architecture?

A Technical Example

While accessing data in new, cloud based platforms becomes more standardized and open with every new release, they are often based on an entity driven API layer based on the OData standard. Dynamics 365 in that respect is no different. It offers direct entity access for both retrieval of data and performing transactions.

One issue however, is the distinction between accessing entities, like customers or invoices, and performing business services. For instance, creating an order might require you to first create the customer, if it does not exist yet, then create the order. In the OData driven API layer this would require you to orchestrate between at least 4 API calls because an order is not even an isolated entity but requires access to order header and order line entities.

This fact requires a customer to create an architecture that, although utilizing standard API's, is about business (micro)services supporting operations across multiple entity services or API's.

Moving to Dynamics 365 is therefore not just about changing your system of record for a large amount of your core business processes; it might also be about recognizing the move from a more passive, polling-for-changes architecture, to a proactive event driven and API enabled architecture.

This also raises questions about availability, load-leveling, monitoring, recovery of transactions and managing overall consistency throughout the application landscape. How will you keep track of events and messages and where can you find information on the state of a single flow of events and/or messages

The Codit Approach

Codit, as an experienced integrator whose sole focus is on creating flexible but robust integration solutions, can help customers and implementation partners in both establishing the to-be architecture and assist in the transition to this target architecture.

As a Microsoft Gold Partner for Integration and an experienced partner in connecting applications in Hybrid and Cloud-Only environments, Codit is uniquely positioned to help customers, not only create flexible and robust integrations, but to also do this in a standardized and accelerated way by making use of our Invictus Integration Framework for Azure and/or BizTalk Server.

While this framework offers standardized monitoring and configuration of integrations, it is also a way of thinking and a way of working. It is the supporting framework that transforms a sound and well thought off integration architecture into high-quality solutions.

Using a proven workshop driven approach we not only pay attention to the mere functional and technical bits, but focus strongly on architectural and other non-functional requirements like monitoring, auditing, alerting, reporting, error handling/recovery and operational support. This enables us to exactly determine how Invictus can help customers best and to start implementations based on a constant quality from the first day of development.

We look forward to meeting you to discuss your future of integration!

To Summarize

This white paper will offer you valuable technical insights into integrating with and accessing data from Dynamics 365 using a variety of proven products, tools and components available from Microsoft. Examples of these include Microsoft BizTalk Server, Microsoft Flow, Azure Service Bus and Logic Apps, and hybrid scenario's where both Azure components and BizTalk Server are used. After reading this white paper you will be able to make a conscious decision on which integration platform to use and how to integrate with Dynamics 365. But...

don't forget the architecture!

1.1 Scope

This white paper offers a guideline for integrating with Dynamics 365. The document is scoped to integrating with Dynamics 365 using BizTalk Server, Azure Service Bus & Logic Apps and Microsoft Flow. Batch jobs handling large data loads and initial import of data into Dynamics 365 are not in scope.

1.2 Intended Audience

We encourage anyone interested in integration and specifically integrating with Dynamics 365 to read this white paper. Although section of the white paper may be fairly technical and therefor more suitable for integration professionals, architects and developers.

1.3 Future Versions

We've already learned quite a bit on projects we've delivered so far, but we keep learning daily! We already have some additions that we plan to release in a 1.1 version of this white paper. If you would like us to keep you informed of news surrounding this white paper, please visit www.codit.eu/blog/dynamics-365-integration/.

2. Defining Our Approach

Instead of beginning from the technical capabilities of Dynamics 365 and mapping them to integration platforms, we instead define several integration patterns which we map to the capabilities of Dynamics 365 and our integration platforms. For an optimal approach, it is best to map these integration patterns to one or more of your business processes. For example, the integration pattern described in chapter 3.2 "Providing Data" might be part of your business process "Create New Employee". This will enable us to determine which integration patterns are relevant for your business based on your business processes.

After defining which integration patterns will be used in your business processes, we will look at the integration capabilities of Dynamics 365. Each way of integrating with Dynamics 365 has its benefits and challenges and by mapping these to our relevant integration patterns you will start to see the bigger picture in relation to your integration architecture.

Next, we'll look at the available integration platforms and its hybrid implementations. If you have an existing integration platform this may be an easy choice, however, especially in greenfield environments you may feel overwhelmed by the available platforms and their capabilities. Based on the relevant integration patterns and the preferred way of integrating with Dynamics 365 we defined earlier, we can make an informed decision on which integration platform and which method of integration will offer the most value for your business.

3. Integration Patterns

3.1 Overview

In the following section, we'll define some common integration patterns when dealing with integration with Dynamics 365.

3.2 Providing Data

By providing data, we mean the process of creating or changing the data in Dynamics 365. The integration platform supplies new or changed data to Dynamics 365 through a synchronous call. The trigger for the process lies outside of Dynamics 365.

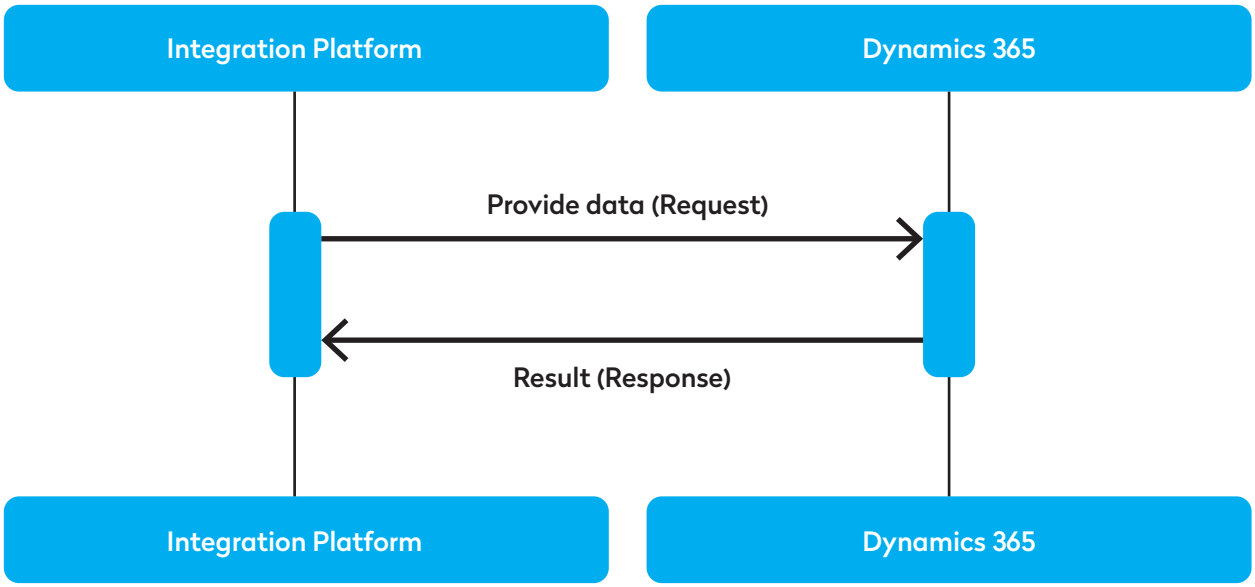


Figure 1: Providing Data Integration Pattern

3.3 Retrieving Data

By retrieving data, we mean the process of getting data from Dynamics 365 where the trigger for the process lies outside of Dynamics 365. The integration platform queries Dynamics 365 for data through a synchronous call.

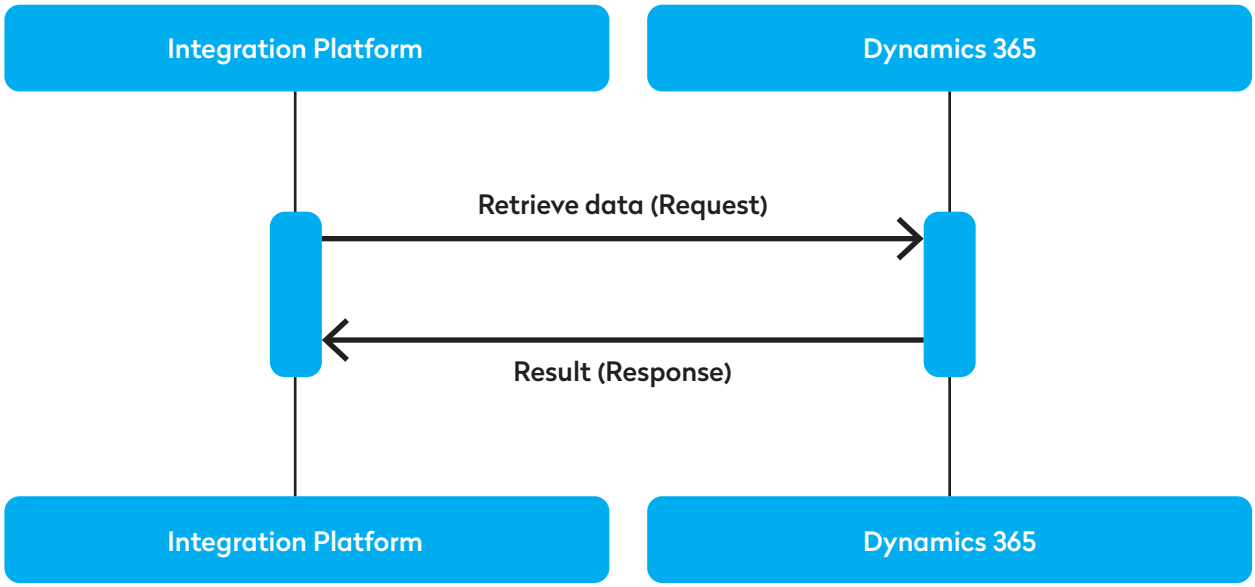


Figure 2: Retrieving Data Integration Pattern

3.4 Receiving Data

By receiving data, we mean the process of getting data from Dynamics 365 where the trigger for the process lies within Dynamics 365. Dynamics 365 supplies new or changed data to the integration platform, usually through an asynchronous call.

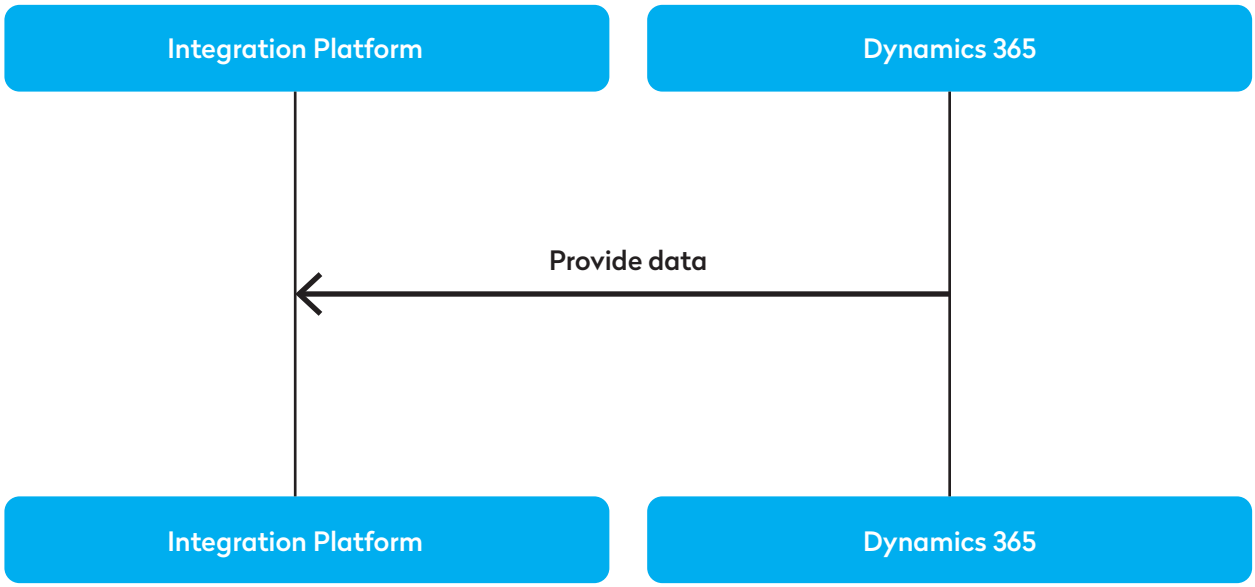


Figure 3: Receiving Data Integration Pattern

4. Integration Capabilities Of Dynamics 365

4.1 Overview

T these are the most common options of integration for Dynamics 365. The benefits, challenges, and risks of each option are described in the paragraphs below.

4.2 Web API

Dynamics 365 comes with a REST Web API which provides a way to interact with the data in Dynamics 365. This Web API makes it possible to create, retrieve, update and delete entities, for a full list of operations see: https://msdn.microsoft.com/en-us/library/mt607901.aspx. These operations provide a rich set of options and allow for most of your common integration needs.

The biggest advantages of the Web API is that it uses industry standards and it's available out-of-the-box and quite easy to use without the need for custom development in Dynamics 365.

While that sounds great, using a Web API also has some drawbacks, as the Web API only supports the "Providing Data" and "Retrieving Data" integration patterns described in chapters 3.2 and 3.3. While this isn't an issue when creating or updating entities, consider a use-case where we must get newly created sales orders from Dynamics 365 to be processed in downstream applications. Continuously asking Dynamics 365 if there are any new sales orders through the Web API is not an ideal solution.

As the Web API supports create, retrieve, update and delete functionality on an entity level, implementing a complex business scenario can result in a process where multiple separate Web API calls are needed. This can result in long-running 'chatty' interfaces and complex error handling. Therefore the "Receiving Data" integration pattern is preferred here where Dynamics 365 sends each sales order to the integration platform directly after being created.

For these scenario's a custom service developed in Dynamics 365 is a better choice, more about this option in chapter 4.3.

4.3 Custom Service

In certain cases, the out-of-the-box Web API cannot offer the required functionality or the message flow would needlessly get too complicated.

Think of a scenario where multiple actions must be taken within Dynamics 365 to process a message while functionally these actions are all part of the same transaction. Using the Web API would mean executing multiple calls to Dynamics 365 and introducing complexity in the error handling, as you must now consider how to resume the message flow if it fails halfway through. A better solution would be to create a custom service in Dynamics 365 which will execute these actions itself.

A custom service can be created to fit an exact need and can be very useful to keep message flows as simple as possible. Do keep in mind that the reusability of these custom services is limited and that these custom services need to be developed on the Dynamics 365 platform, this means that development of these services is done outside of the integration team. From a project management standpoint, the need

for a custom service in Dynamics 365 needs to be identified early on so that the implementation partner for Dynamics 365 can get involved as soon as possible for the development of the custom service.

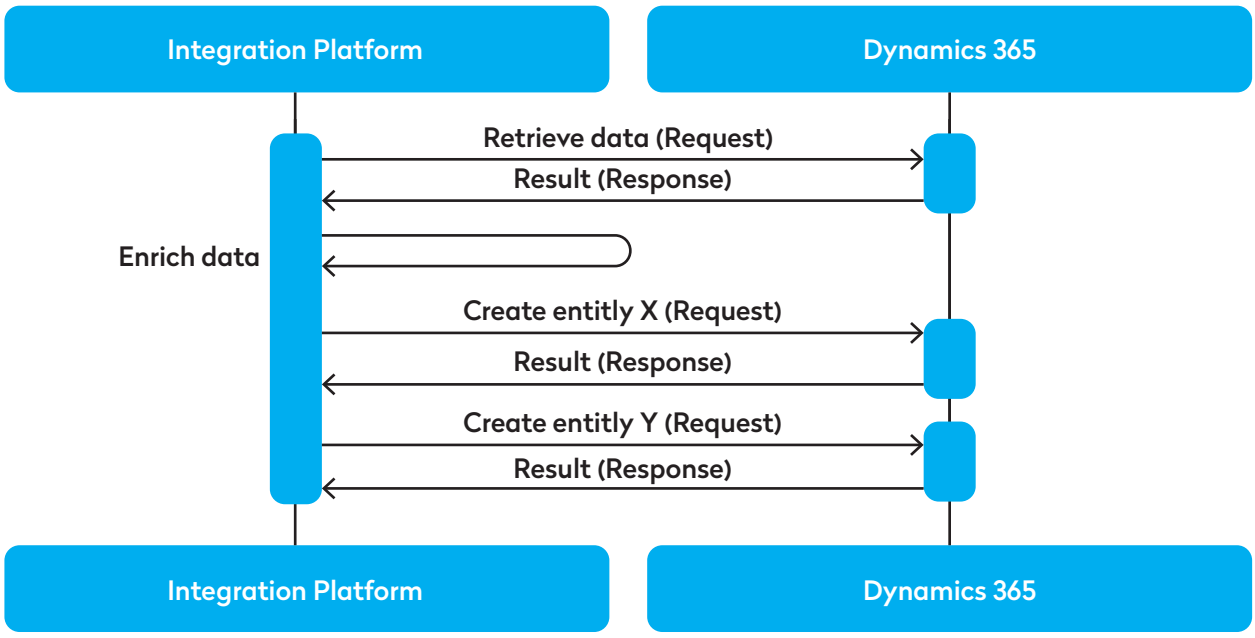


Figure 4: The example scenario shown in the Web API solution

As shown in figure 4, using the Web API solution for our example scenario contains multiple service calls from our integration platform to Dynamics 365. This requires complex error handling in our integration platform and can negatively impact the performance of the interface.

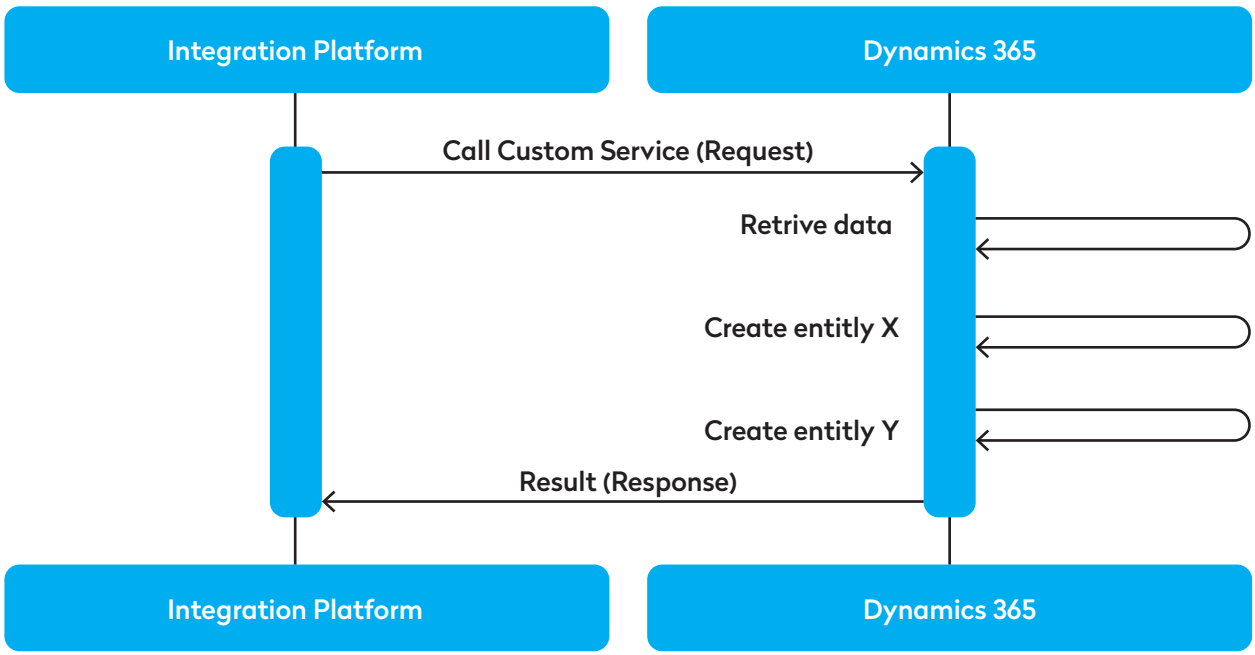


Figure 5: The example scenario shown in the custom service solution

As shown in figure 5, using the custom service solution in the example scenario contains one service call from our integration platform to Dynamics 365. As a result, error handling in the integration platform is kept as simple as possible.

4.4 Event Pipeline

When the trigger for an event lies within Dynamics 365, for example, the creation of a sales order, it is more logical to have Dynamics 365 publish the data than to have a client constantly querying if new data is available. To support this functionality, it is possible to use the Dynamics 365 event execution pipeline and, for example, call a BizTalk endpoint or connect with Azure Service Bus and place the message on a queue or topic. For this, custom plug-ins must be developed and registered in Dynamics 365. This approach enables the "Receiving Data" integration pattern described in chapter 3.4.

As with the custom service described in the previous chapter, the plug-ins need to be developed on the Dynamics 365 platform, meaning the development is done outside of the integration team and thus comes with the same consequences regarding project management.

4.5 Common Data Service & Data Integration

The Common Data Service is a data storage platform in PowerApps where entities can be stored in a Common Data Model. From here the data can be accessed from multiple applications such as the Dynamics 365 family of services, Office 365 and Power BI. The implementation of the Common Data Model results in a consistent way of defining your business entities across applications. For example, multiple applications might have the same entity called "customer", but it has a different definition in each system. By using Common Data Service, these different definitions can be mapped to one single "customer" definition in the Common Data Model. Now, when the data for a customer gets updated in the Common Data Model, all connected applications are updated.

Dynamics 365 Data Integration enables the flow of data across the Dynamics 365 family of services, moving the data manually or using a separate integration platform to do this is no longer necessary. Furthermore, no custom code is needed to set up Dynamics 365 Data Integration; it is all done via configuration in PowerApps.

Do keep in mind that the use of Common Data Service and Data Integration is only applicable when using more than one module in Dynamics 365, for example, Dynamics 365 for Sales and Dynamics 365 for Finance and Operations.

5. The Integration Platforms

5.1 Overview

This chapter focuses on the different integration platforms that Microsoft offers.

5.2 BizTalk Server

BizTalk Server is the most mature product on the list, it has been around since the early 2000's. BizTalk Server can easily consume any of the Dynamics 365 endpoints, whether it is REST or SOAP. It can also support any of the integration patterns we identified in chapter 3. It should be noted, that to implement the "Receiving Data" pattern described in chapter 3.4, custom code needs to be developed in Dynamics 365 to send the data directly to BizTalk Server.

There are some technical challenges here, as Dynamics 365 endpoints are authenticated using OAuth 2.0, which is not supported by BizTalk Server out-of-the-box. However, because of the extensibility options that BizTalk Server offers, it possible to use a custom WCF Behavior to add this functionality.

BizTalk Server is currently available as both an on-premise platform or as an Azure Infrastructure as a Service (IaaS) platform.

5.3 Azure Service Bus & Logic Apps

Dynamics 365 offers support for integration using Azure Service Bus and Logic Apps.

Service Bus provides decoupling through reliable message queueing and durable publish-subscribe messaging and contains queues, topics, and relays. Describing these components in short:

- Queues are used to store messages and are received by a single recipient.
- Topics, like queues, are used to store messages but can have multiple recipients through different subscriptions. Also, optionally only messages that match certain criteria can be received through a filter on the subscription.
- A relay provides two-way communication, messages are not stored but simply passed through to the destination service.

Azure Logic Apps provide a way to implement integrations, automate workflows and communicate with services and applications through its many available connectors.

To implement the "Receiving Data" pattern, as described in chapter 3.4, Dynamics 365 can directly write to queues or topics in Azure Service Bus. To enable this, a custom plug-in must be developed. Another option to implement this integration pattern is by using Logic Apps. Logic Apps provide a set of connectors to Dynamics 365, with these connectors you can:

- Create a new record
- Delete a record
- Get a record
- List records
- Update a record
- Get a notification when a record is created
- Get a notification when a record is created or updated
- Get a notification when a record is deleted
- Get a notification when a record is updated

There are also Logic App connectors for the Common Data Service, which are only relevant when Dynamics 365 Data Integration is used. With these connectors you can:

- Create a new record
- Delete a record
- Get a record
- List records
- Update a record
- Generate a key value
- Get a notification when a record is created
- Get a notification when a record is updated

For a complete list of connectors and their description see
https://docs.microsoft.com/en-us/connectors/crm/ and
https://docs.microsoft.com/en-us/connectors/runtimeservice/.

The “Get a notification” and “When a record is created/updated” connectors are polling based, meaning that you must supply an interval on how often you want to check for the specific event. Keep in mind that every time the Logic App is triggered Azure consumption costs are incurred, so setting the interval to 1 second will cost considerably more than setting it to 1 minute. For the current Logic App pricing details, see https://azure.microsoft.com/en-us/pricing/details/logic-apps/.

As you can see, these connectors cover all the integration patterns described in chapter 3. It is also important to note that, as opposed to BizTalk Server, no custom coding is needed to perform the required authentication with Dynamics 365.

5.4 Microsoft Flow

With Dynamics 365 it’s now possible to integrate using Microsoft Flow. Flow is built on top of Logic Apps and they share the same workflow designer and connectors. Where Logic Apps are used for advanced integration scenarios by integration specialists, Microsoft Flow can be used by business users to automate simple workflows. However, Logic Apps has a richer ALM experience, offers greater control for administrators and more security capabilities.

As Microsoft Flow is built on top of Logic apps and contains the same connectors, the functionality as described in chapter 5.2 applies here as well. With Microsoft Flow, you can implement all of the integration pattern described in chapter 3.

6. Hybrid Implementations

This chapter considers the options that are most relevant for organizations that are more likely to spend a longer period of time in a hybrid application landscape, where they will be using core systems like Dynamics 365 in the cloud, but will also continue using on-premise applications.

This option also works well for organizations that have already invested in BizTalk Server integration solutions in the past, as this option allows them to leverage previous investments and use BizTalk Server on-premise and Azure integration components side-by-side.

6.1 Combining Azure Service Bus And BizTalk Server

In this setup we use Azure Service Bus queues to receive data from Dynamics 365, the data is then consumed by BizTalk Server from the message queue and processed. A plug-in must be developed in Dynamics 365 to place the data on the queue. This implements the "Receiving Data" pattern described in chapter 3.4.

To implement the "Providing Data" and "Retrieving Data" described in chapter 3.2 and 3.3, BizTalk Server will directly call Dynamics 365.

This option allows organizations to maximize reuse of existing knowledge of BizTalk Server, as nearly all integration artifacts will continue to be built in BizTalk Server, while Service Bus is only used for message transport.

An additional benefit of this approach, compared to only using BizTalk Server, is that the need for high/always available endpoints in BizTalk Server may become less critical. If, for example, BizTalk Server is down for maintenance, the messages will remain in the Azure Service Bus queue until BizTalk Server is up again.

6.2 Combining Azure Service Bus, Logic Apps and BizTalk Server

In this scenario, we consider using the Logic App connectors to handle all communications with Dynamics 365. To implement the "Receiving Data" pattern described in chapter 3.4 the Logic App connectors will be used to receive a notification on an event and publish the data on an Azure Service Bus queue, the data is then consumed by BizTalk Server from the message queue and processed.

To implement the "Providing Data" and "Retrieving Data" described in chapter 3.2 and 3.3, BizTalk Server will call a Logic App which will take care of the authentication and call Dynamics 365.

As in the previous chapter, by using this approach compared to only using BizTalk Server, we may minimize the need for the endpoints in BizTalk Server to be high/always available. If, for example, BizTalk Server is down for maintenance the messages will remain in the Azure Service Bus queue until BizTalk Server is up again.

However, contrary to what we said in the previous chapter when using this approach, no custom code is needed in Dynamics 365.

6.3 Including API Management

API Management can be used to completely technically separate Dynamics 365 from the integration platform. The API Management layer can be added to any of the communication platforms and the hybrid scenarios described earlier. Such a layer also enables throttling scenarios, protocol mediation, configurable policies, caching and security, as well as monitoring.

API Management can be used bidirectionally, as in communicating both to and from Dynamics 365, covering all integration patterns defined in chapter 3. However, to implement the "Receiving Data" pattern, it is necessary for custom code to be developed in Dynamics 365 to call the API's provided by the API Management layer.

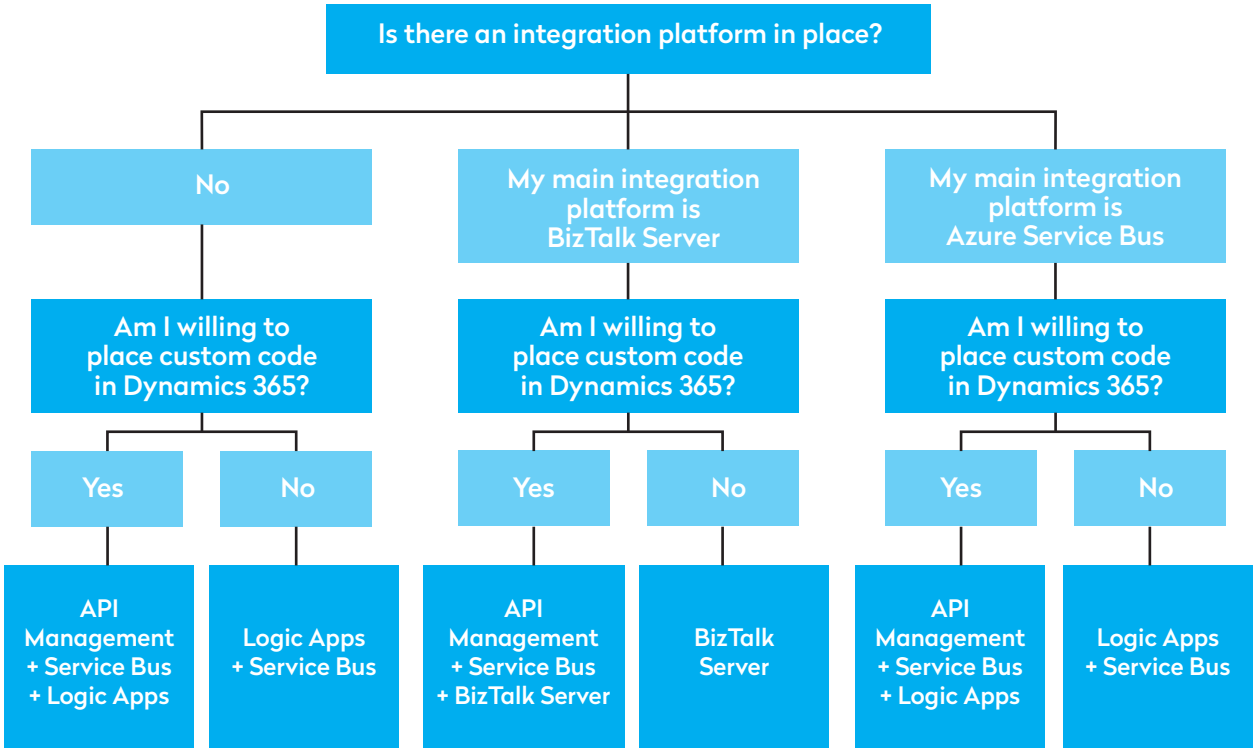
7. Choosing an Integration Platform and Method

Now that we have described the integration capabilities of Dynamics 365, the available integration platforms and their hybrid implementations, let's try to define which integration platform and which method of integration is best for which scenario.

7.1 The Platform

Firstly, its important to make the distinction between Dynamics 365 hosted in the cloud and Dynamics 365 hosted on-premise. When Dynamics 365 is hosted on-premise and will not be moved to the cloud in the near future, choosing an integration platform is straightforward. The choice here is to go with BizTalk Server, as described in chapter 5.1, as the other options are focused on the cloud.

When Dynamics 365 is hosted in the cloud the decision becomes a bit more complex and can be determined by asking the following questions.



There are of course some nuances. If your main integration platform is BizTalk Server and you are not willing to place custom code in Dynamics 365, the outcome is to use BizTalk Server. However, if you are looking into moving your integration platform to the cloud, a good first step could be to combine BizTalk Server and Logic Apps, by using the Dynamics 365 authentication available in Logic Apps.

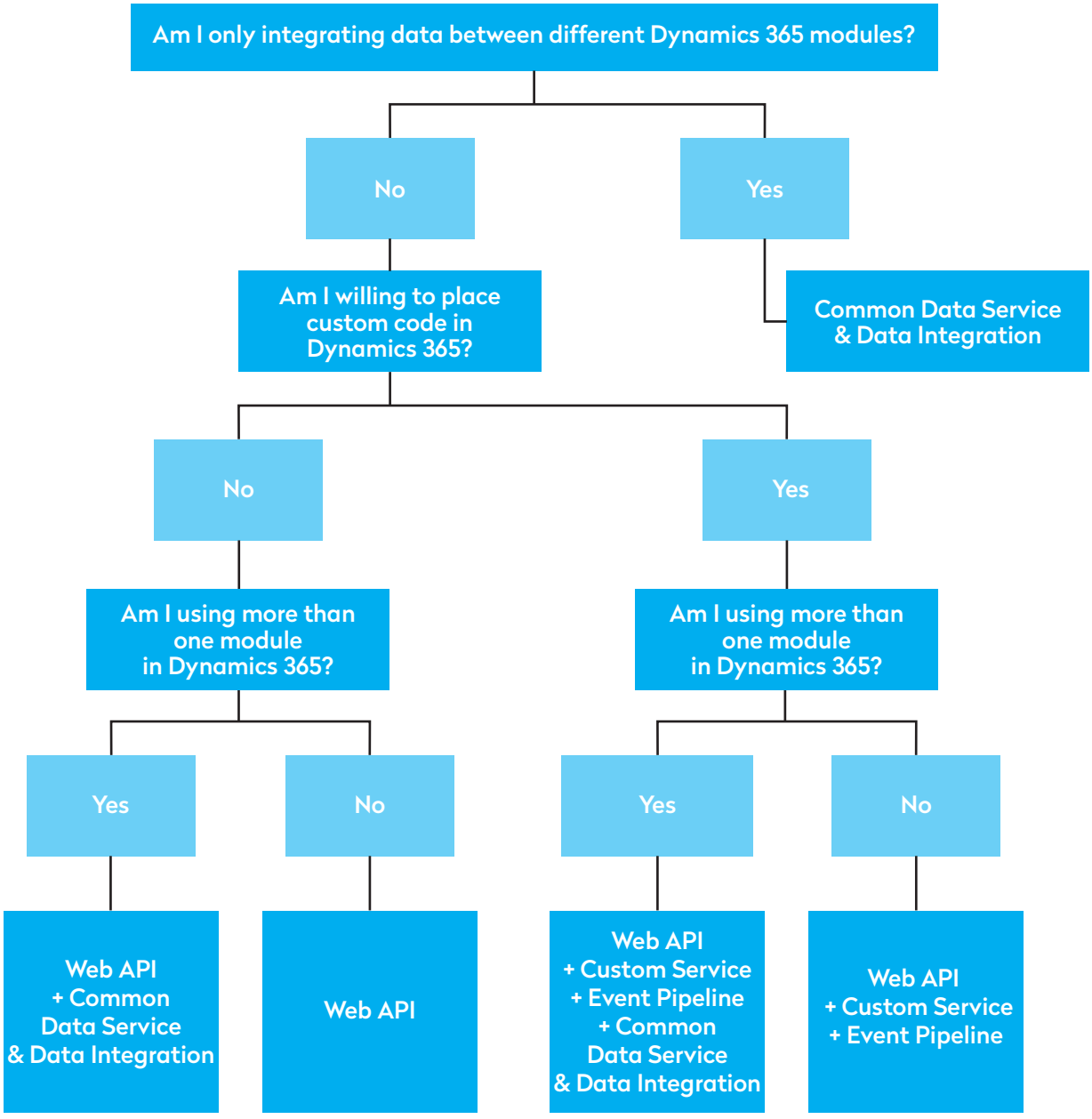
The question "Am I willing to place custom code in Dynamics 365?" is an important one, without custom code in Dynamics 365, it will not be possible to call custom endpoints made available in API Management. There are valid reasons for not wanting custom code in Dynamics 365, such as not wanting to burden administrators with maintaining custom code and wanting to avoid the possibility of updates to Dynamics 365 causing broken custom code.

Whether you use BizTalk Server or Logic Apps and Service Bus as your integration platform, generally the integration architecture should stay the same. We advise using API Management as a layer between Dynamics 365 and your integration platform to separate the platforms.

7.2 The Method

We'll now discuss the method of integrating with Dynamics 365. As described in chapter 4, there are several options available. As with the decision for the integration platform, the distinction between Dynamics 365 hosted in the cloud and Dynamics 365 hosted on-premise is an important one. When Dynamics 365 is hosted on-premise, the Common Data Service & Data Integration option is not available.

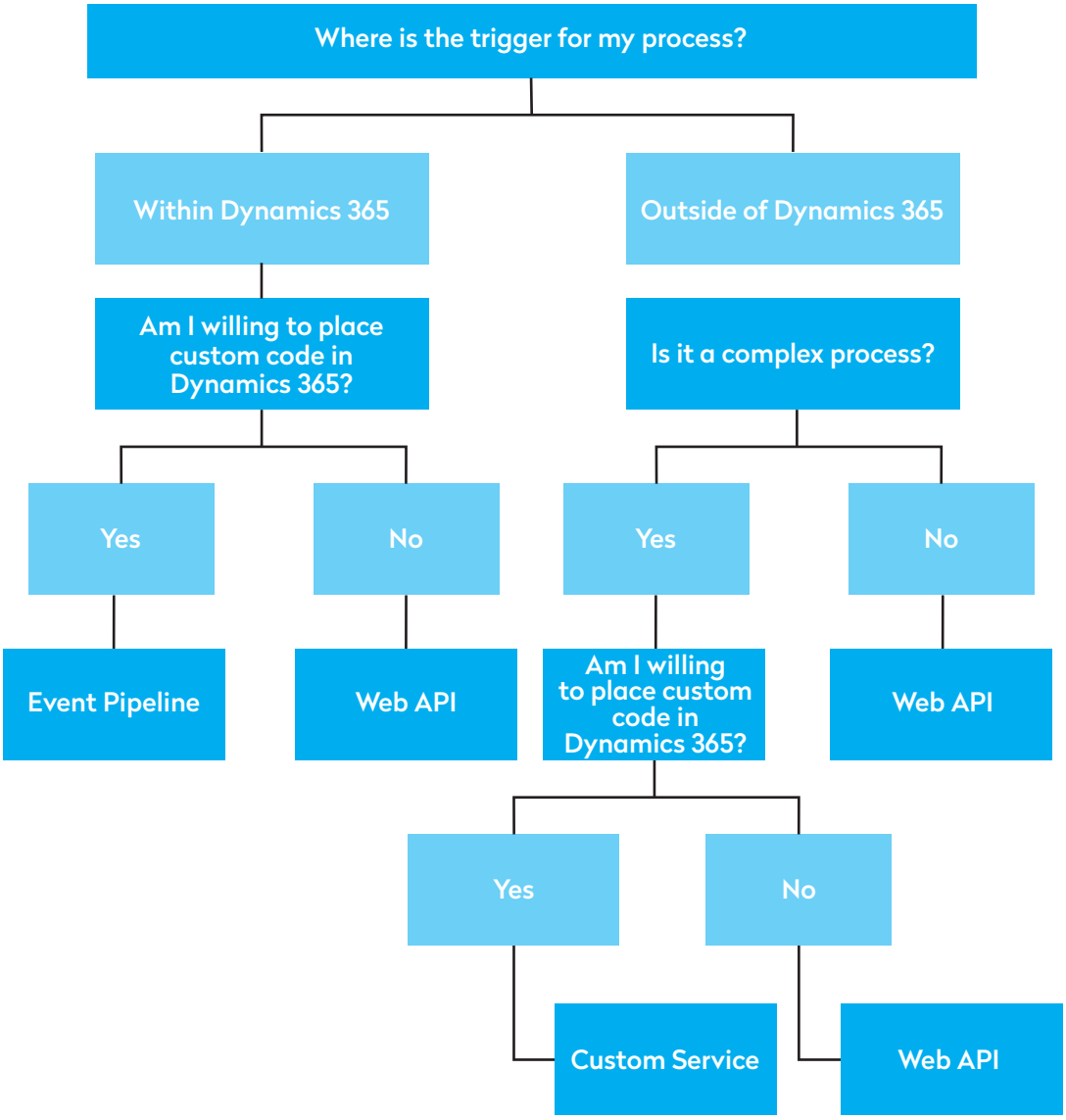
When Dynamics 365 is hosted in the cloud the **methods of integrating** available to you can be determined by asking the following questions.



Again, the question "Am I willing to place custom code in Dynamics 365?" is an important one, without custom code in Dynamics 365 it will not be possible to use the Event Pipeline and Custom Service. As described earlier, there are valid reasons for not wanting custom code in Dynamics 365, such as not wanting to burden administrators with maintaining custom code and wanting to avoid the possibility of updates to Dynamics 365 causing broken custom code.

The Common Data Service and Data Integration option is an intriguing one, but at the moment of writing are really only applicable to the integration of data amongst the different modules of Dynamics 365.

If the outcome of the decision tree above comes down to a combination of the Web API, Custom Service and Event Pipeline we still need to decide when which of the methods offers the best solution. You can use the following decision tree to determine **per interface** which of these methods of integrating is the one to use.



8. Conclusion

We have focused on the integration capabilities of Dynamics 365, the available integration platforms, its hybrid implementations and how to choose an integration platform and method. The integration capabilities of Dynamics 365 has been expanded. While this introduces many new **opportunities**, it also poses new **challenges** and questions around which platform and which method of integration to use.

Based on our experience integrating with Dynamics 365, feedback from our customers and from Dynamics 365 implementation partners, we have narrowed down the different choices and decisions you will have to make. We have visualized them in the decision trees described in chapter 7. With these decision trees, we hope we can provide you guidance on how to set up your integration solutions around the addition of Dynamics 365 into your application landscape, assuming you have an integration architecture in place.

One more piece of advice, don't be overwhelmed by the integration capabilities of Dynamics 365 and integration platforms at your disposal. Dynamics 365 is a great product, and the integration capabilities we now have available are very exciting, don't be afraid to integrate with Dynamics 365!

If you are interested to talk to Codit about your specific integration requirements and/or are seeking a partner to help you on your integration journey please contact your local Codit office which can be found at https://www.codit.eu/en/contact/.

9. References

Use Microsoft Dynamics 365 web services:
https://msdn.microsoft.com/en-us/library/mt608128.aspx

Perform operations using the Web API:
https://msdn.microsoft.com/en-us/library/mt607901.aspx

Connect to Microsoft Dynamics 365 web services using OAuth:
https://msdn.microsoft.com/en-us/library/gg327838.aspx

Azure integration with Microsoft Dynamics 365:
https://msdn.microsoft.com/en-us/library/gg334766.aspx

Connect to Dynamics 365 from Logic App workflows:
https://docs.microsoft.com/en-us/azure/connectors/connectors-create-api-crmonline

Dynamics 365 Logic App Connector:
https://docs.microsoft.com/en-us/connectors/crm/

Dynamics 365 Data Integration:
https://docs.microsoft.com/en-us/common-data-service/entity-reference/dynamics-365-integration

Introduction to Data Integration with Common Data Service:
https://docs.microsoft.com/en-us/Dynamics365/unified-operations/dev-itpro/data-entities/data-integration-cds

Message Queueing in Dynamics 365 with Azure Service Bus:
https://community.dynamics.com/enterprise/b/xrmandbeyond/archive/2017/11/11/message-queueing-in-dynamics-365-with-azure-service-bus

Notes

codit

About Codit

Codit is an innovative IT company which provides next-level consultancy, technology, and managed services to leading brands worldwide. We successfully help companies reduce operational costs, improve efficiency and enhance communication by integrating people, applications, and things.

Codit employs more than 180 people in Belgium, France, Portugal, Switzerland, United Kingdom, The Netherlands, and Malta. Since 2000, we have successfully implemented over 500 integration solutions worldwide.

info@codit.eu | twitter: @CoditCompany | **codit.eu**