A picture containing drawing, food

Description automatically generated

BrewTech Solutions

Trade Off Analyse

Raf Vanhoegaerden

Contents

[1 Context 2](#_Toc1975007193)

[1.1 As-Is 3](#_Toc226175286)

[1.2 To-Be 3](#_Toc1086855191)

[1.3 Solution 3](#_Toc496703565)

[2 Trade off analysis 3](#_Toc1792274126)

[2.1 Performance 4](#_Toc1703348411)

[2.1.1 External Hook 4](#_Toc585669252)

[2.1.2 Autoflow 4](#_Toc1853506055)

[2.2 External Job 4](#_Toc1862801486)

[2.3 Accuracy 5](#_Toc431615821)

[2.3.1 External hook 5](#_Toc561225304)

[2.3.2 Autoflow 5](#_Toc1247162601)

[2.3.3 External job 5](#_Toc290642446)

[2.4 Complexity 5](#_Toc1434653151)

[2.4.1 External hooks 6](#_Toc722491250)

[2.4.2 Autoflow 6](#_Toc1017193489)

[2.4.3 External job 6](#_Toc772612544)

[2.5 Scalability 6](#_Toc31850732)

[2.6 Cost & Resources 7](#_Toc2133037978)

[2.7 Propose the Best Fit 7](#_Toc1498427233)

[3 Conclusion 8](#_Toc1947794234)

# Context

## As-Is

At BrewTech Solutions, maintaining accurate and up-to-date customer and prospect data is a priority for the sales team. Ensuring the correctness of contact information is critical to targeting the right individuals within customer and prospect organizations.

In SAP Sales Cloud V2, this data is managed within the ‘Accounts‘ section, each account has a main contact that is visible through the relationship between the account and its associated contact person. When multiple contact persons are linked to an account, one can be designated as the main contact. This selection updates the account’s header-level information.

However, a key limitation has been identified. Out-of-the-box functionality does not provide a visible indication on the ‘Contacts’ screen to identify which contact is marked as the main contact for a linked account.

## To-Be

When the main contact relationship is changed at the account level, the system should call an external service to automatically update the “MainContact” field on the linked contacts and update the relationship at the account level.

When there is already a main contact relationship at account level and we want to change the main contact, the system must know which contact was the main contact before for the account and update the “MainContact” field accordingly of the contact.

## Solution

SAP Sales and Service Cloud V2 provides different functions to tailor the behavior of the applications to your needs. Especially Determinations, Validations and **Autoflows** allow you to create rules, based on changes to objects in the system.

The build in capabilities is limited by purpose, to have a clear separation between configuration and custom code. If the intended scenario is exceeding the build in functionality, SAP Sales and Service Cloud V2 is providing webhook based extensions points for custom code.

In SAP Sales and Service Cloud V2, we distinguish between **asynchronous** and **synchronous** webhooks. Synchronous webhooks intercept the operation and provide the capability to react or change direct user input. Asynchronous webhooks are called after the operations are finished. They allow to trigger processes on changed data. It is important to keep in mind that synchronous webhooks are blocking the execution of a request.

# Trade off analysis

## Performance

### External Hook

Performance is key for making things run smoothly. The external hook implementation should be very lean, efficient, and performant as these are called **synchronously** within the transaction. Sub-optimal implementations hamper the performance of your application.

This means that the user needs to wait until the transaction receives a **response** from the external application, which could lead to potential delays. The time taken for an external hook response directly impacts the user experience and system efficiency. To ensure a high performance additional measures like caching might be recommended. If the response time is too long, it may result in time loss and possible operational bottlenecks.

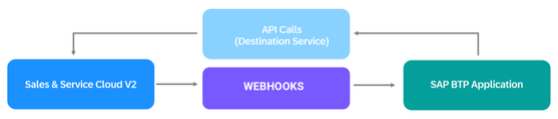
In unforeseen cases, to protect the system's stability and performance, requests taking more than 10 seconds are terminated. An error message indicating this appears in the application.



### Autoflow

This method relies on having our v2 system triggering a webhook, which is an HTTP request, sent to the external service, containing a data payload, which triggers certain action **asynchronously**.

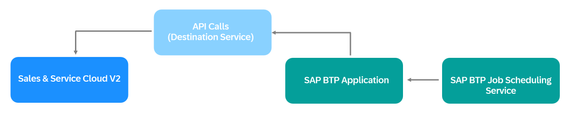
Autoflow is an asynchronous execution which means there is no impact on end-user's performance, the application has "no rush” to do what it needs to do as per this asynchronism. As per this method, even in case the external service fails, users could continue with their daily tasks without these being affected.



## External Job

This overall concept is to use the **BTP job scheduling service** to trigger certain activities. Alternatively, some CPI flows could be also scheduled. This is useful when needing to adjust information under certain conditions or under certain periods of time, when not able to use scheduled field updates or autoflows.

It’s an asynchronous execution which means there is no impact on end-user's performance. Which is an important detail, because even if the external service fails, users could continue with their daily tasks without these being affected.



## Accuracy

### External hook

External hooks operate synchronously, ensuring that data updates occur immediately and in real-time. This guarantees that the "MainContact" field is updated correctly before the transaction is completed, reducing the risk of outdated or conflicting data. However, if the external service fails or responds incorrectly, there is an immediate impact, possibly blocking the transaction or requiring manual intervention.

### Autoflow

Autoflow operates asynchronously, meaning updates to the "MainContact" field happen after the transaction is completed. This ensures that users can continue their work without delays, but there is a small risk that the update might not occur immediately. If the external service fails or delays in processing the request, there could be a temporary inconsistency in the data, requiring monitoring or retry mechanisms.

### External job

Since external jobs are triggered periodically, data updates may not happen immediately but will be consistently applied based on the schedule. This method ensures reliable updates while avoiding real-time system load.

Because updates are executed based on a predefined schedule, there is a risk of temporary inconsistencies. If a user makes a change to the main contact, it might not be reflected in the system immediately. The time of the temporary inconsistency is defined by how often the schedule is set to execute the job. The more often the job is executed, the higher the load but the lower the data inconsistency. Since the updates happen in bulk, failures can affect multiple records at once.

## Complexity

### External hooks

Requires synchronous execution, which means careful handling of response times and failure scenarios to avoid blocking transactions. Implementing and maintaining external hooks demands strict error handling and optimization to prevent performance bottlenecks. Debugging issues can be challenging as failures directly impact user operations, requiring immediate troubleshooting.

Suitable for straightforward data updates but becomes complex when dealing with multiple dependent updates or external system dependencies.

### Autoflow

Easier to implement as it runs asynchronously, reducing immediate system dependencies. Allows for better fault tolerance since failures do not block the main transaction, but requires a retry or monitoring mechanism to ensure updates occur. Debugging can be more complex due to the delayed execution, tracking and tracing changes requires additional logging mechanisms.

Suitable for scalable and long-running processes but might introduce complexity when ensuring real-time consistency.

### External job

Moderate complexity, while setting up job scheduling services (such as SAP BTP job scheduling or CPI flow triggers) requires initial configuration, the ongoing maintenance and monitoring are generally simpler than real-time hooks.

Unlike Autoflow, which automatically triggers updates based on events, External Jobs require manual schedulingand configuration. This includes setting the right frequency, defining execution conditions, and ensuring that jobs do not interfere with other scheduled tasks in the system.

While External Jobs can efficiently handle large volumes of data, they require **careful optimization** to prevent system strain. Poorly managed jobs could lead to performance bottlenecks, where updates take longer to process as data volume grows.

Ensuring that the correct records are updated without duplication or omission requires additional validation mechanisms. This adds another layer of **complexity to error handling**, especially in cases where multiple concurrent updates are being processed.

## Scalability

**External Hook** is suitable for small to medium workloads where immediate updates are necessary but can struggle with scalability as load increases. Limited flexibility since it requires synchronous execution, making it harder to adapt to new workflows without affecting system performance. It’s important to know that with this method, we can only use one external pre hook and one external post hook. So, for further scalability it will be difficult to further extend the functionality, if needed in the future.

**Autoflow** provides better scalability by decoupling data processing from user transactions, making it more robust for large-scale operations. More adaptable as it runs asynchronously, allowing for easier adjustments to workflows without impacting system transactions.

Autoflow is more future proof, allowing seamless integration of new processes without affecting system performance, making it the better choice for scalability and adaptability.

In both cases, due to the open architecture of SAP Sales and Service Cloud V2, it is possible to choose the right architecture, which not only solves the technical requirements but also fits with the strategy of the company. Instead of using Node JS and Cloud Foundry it would be possible to use Java. The only requirement is the endpoint must be available through the internet and listen to HTTP Post requests.

**External jobs** are highly scalable as jobs can be adjusted in frequency and volume depending on the business need. They are ideal for large-scale updates, periodic data synchronization, and background processing tasks.

## Cost & Resources

Both **External Hook** and **Autoflow** require the development of an external service, along with considerations for memory usage. The cost and resource requirements for these two methods are nearly identical.

We will leverage SAP BTP services, including an **autoscaler** and an **authorization service**, both of which are free. However, for memory usage, we will need a **Cloud Foundry runtime**. In this context, our estimated memory requirement is around **0.5 GB**. To put this into perspective, 1 GB of memory in BTP costs approximately €85 per month, meaning the cost for these methods will be roughly **€40 per month**.

**External Job** also requires the development of an external service but comes with an additional cost due to the need for a job scheduling service. However, because we rely on scheduled executions rather than continuous memory usage, overall memory consumption will be lower.

For an **External Job**, we need to incorporate a job scheduling service, which costs around **€14 per month** for an execution block of **10,000 jobs**. To provide some perspective, the minimum scheduling interval is five minutes. If we schedule a job every five minutes, 24/7, for a full 30-day month, we will execute **8,640 jobs per month**, keeping us within this cost bracket.

Overall, while **External Job** incurs slightly higher costs due to the scheduling service, the differences remain minimal across all three methods.

## Propose the Best Fit

Based on the trade-off analysis of **External Hook,** **Autoflow** and **External Job**, the best fit depends on the business priorities of **performance**, **accuracy**, **complexity**, **scalability**, **and cost & resources**. After a thorough evaluation of the different options.

I recommend the **Autoflow** method as the best fit for BrewTech Solutions.

The main advantage of an autoflow is that it operates **asynchronously**, meaning it has no impact on user interface performance. Users can continue their tasks without delays or interruptions. This is crucial in a production environment where speed and ease of use are key priorities. However, we should consider there is a temporary data inconsistency. But this is so small we can neglect that.

While an external hook operates synchronously, meaning the system waits for an external service to respond before proceeding. If the response from the external service takes time, it **directly impacts the user experience**. Also, if the request takes 10 seconds it will be terminated, and the user could face delays or in this case a bottleneck.

An external job would not be ideal for real-time updates in this context. External jobs are meant to run at specific intervals and typically process data in batches. Which could cause too big of a potential delay in data consistency.

While both **External Hook** and **External Job** have specific use cases, **Autoflow** offers the most comprehensive and future-proof solution for BrewTech Solutions. It strikes the right balance between **performance**, **scalability**, **flexibility**, and **cost-effectiveness**, making it the ideal choice to address current and future business needs.

# Conclusion

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **External Hook** | **Autoflow** | **External Job** |
| **Performance** | Synchronous, immediate but may introduce transaction delays | Asynchronous, no impact on user experience | Asynchronous, triggered at scheduled intervals, ensuring minimal system impact |
| **Accuracy** | Ensures real-time updates | Slight risk of temporary data inconsistency | Depends on the scheduling frequency—delayed updates but ensures data consistency |
| **Complexity** | Requires real-time execution, more challenging to debug and maintain | Easier to implement, but requires monitoring for failures | Moderate complexity, as job scheduling requires proper orchestration and handling of dependencies |
| **Scalability** | Harder to modify without affecting system performance | More flexible for future changes and integrations | Highly scalable since jobs can be scheduled dynamically, independent of real-time user interactions |
| **Cost & Resources** | Requires external service development and ~€40/month for memory usage | Similar to External Hook, ~€40/month for memory usage | Requires external service development and job scheduling (€14/month), but lower memory costs |