B2B Platform Federation

Yoav Tock[a], Benjamin Mandler[a], Suat GÖNÜL[b], Doğukan ÇAVDAROĞLU[b], Nir Naaman[a] and Nir Rozenbaum[a]

[a] *IBM Haifa Research Lab, Haifa, Israel*
[b] *SRDC Software Research & Development and Consultancy Corp., Ankara, Turkey*

Abstract

B2B platforms are becoming ever more prevalent, for digitizing supply chain and logistics interactions. One direction is to create a single very large platform, to which all companies registered, and is controlled by a single entity. A different approach calls for small platforms that interact with each other in a federated manner. In this paper we describe the latter manner, in which an existing B2B platform can be federated. A requirement is that all individual platforms joining a federation expose the same set of base interfaces for their core services. In addition, each such individual platform may be tailored to a specific domain or region.

Keywords 1

B2B, federation, supply chain, identity

1. Introduction

The NIMBLE project (see: https://www.nimble-project.org/) aims to perform research leading to the development of a cloud platform specifically targeted to supply chain relationships and logistics. Core capabilities enable firms to register, publish catalogues for products and services, search for suitable supply chain partners, negotiate contracts and supply logistics, and develop private and secure information exchange channels between firms. The intention is to support a federation of NIMBLE instances, all providing a set of core services, and each potentially specifically tailored to a different aspect. The vision is to enable a federation of NIMBLE instances, such that end users belonging to different NIMBLE instances may engage in B2B operations. While each instance provides the core common capabilities as mentioned above, A NIMBLE provider can take the open source infrastructure and bundle it with sectorial, regional or functional added value services and launch a new platform instance. Such specializations may take place at the industry level, namely adding specific capabilities for a specific industry, or at a regional level. NIMBLE aspires to a federated yet interoperable eco-system of platforms that provide B2B connectivity. Such a common, yet federated infrastructure opens the door for multiple platform providers, with a diverse set of platform instances that still can collaborate. This concept provides also the ability of the administrator or governing body of an instance to decide with which other instances to federate.

Prior art includes MEMORAe [1] which follow a Leader/Follower architecture, while our case is symmetric. A challenge in developing federated services is the lack of central control [2]. To address this, we place a delegate service in each instance which registers itself to a centralized identity service and the delegate is the only communication point for instances.

2. Architecture High Level View

The main elements of the proposed federation architecture are the federation core services and the instance delegates as shown in Fig. 1. The federation core services are placed outside of all participating instances and contain general management support for the federation. The instance delegate is placed at the edge of each instance representing that instance in the federation.

Proceedings of the Workshops of I-ESA 2020, 17-11-2020, Tarbes, France
EMAIL: TOCK@il.ibm.com

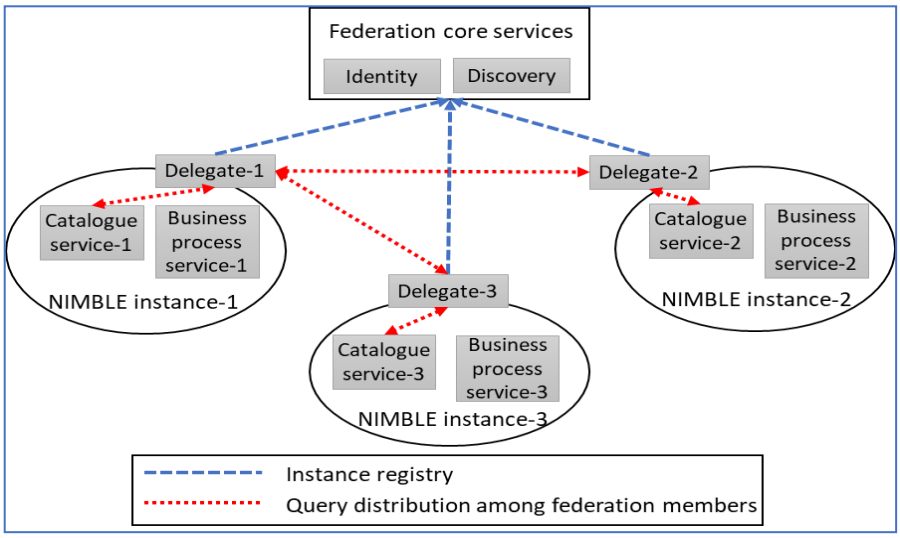

Figure 1: High-level view of the federation architecture

The instance delegate is an entity identified, registered and deployed within an instance. It is also identified and registered in the federation core services. It controls the communication and collaboration patterns between local services (i.e. catalogue service) and external instances. The delegate obtains information from the federation core services concerning its counterparts in other instances and proceeds to invoke the respective operations directly on the delegate belonging to the collaborating instance. The delegate of one instance communicates with the delegate of a second instance, not directly with the local services of the second instance. The interaction is performed using a RESTful API. A local service will contact its local delegate, which will contact the delegate of a remote instance, and that delegate will forward the request to its local services. Responses will be sent back, following the reverse path, to the initiating entity. Responses will be combined by the local delegate that initiated the call.

Security and privacy are the main reasons for placing a delegate at the edge of each instance and allow it to communicate only with the respective delegate of a remote instance. This way, an internal service or entity only communicates with a local trusted component. Each instance administrator can configure and enforce what information stays within the instance and what is permitted to be exposed to the federation. The delegate is a configurable gatekeeper that is used to protect the data of an instance, in addition to its role of connecting it to the federation. The requests flowing in the route service->delegate->delegate->service carry the bulk of the data that flows between instances, and therefore constitute the federation "data-plane", represented with the red lines. It consists mainly of API calls forwarded back and forth between instances.

The federation core services allow NIMBLE instances to federate and end-users to collaborate. The interaction between the federation core services and the delegates can therefore be called the federation "control-plane", represented with the blue lines. These services will provide a set of APIs that allow the federation core to provide the following capabilities: instance administrators to register their platform instance for federation; instance administrators to define identities for delegates to their respective platforms; a delegate to register and identify itself as a representative of an instance; delegates to search for other instances and discover the delegates that represent them.

The identity service is used to establish an identity for an instance and an identity to the delegate that belong to that instance. The discovery service allows delegates to discover other live delegates. A delegate connects to the discovery service and publishes its identity, affiliation, and address. As long as it remains connected it will be kept as "alive" in the discovery service. Other delegates can query the service for all live delegates. In a sense this is a membership service for delegates.

3. Setting up a Federation

The federation components represent the main aspects which are required to take a "standard" instance and allow it to federate with other instances. The first aspect is deploying the federation core services. The second aspect is deploying a delegate in every instance. The delegates are then registered to the federation core, and an administrator configures the roles of each delegate. Some extensions to the capabilities of local services are recommended, for example tagging each item, service or capability with a "scope" of whether externalizing it in a federation is permitted.

There are manual steps which are required before a federation is created. These are not technical steps but rather managerial and organizational steps. First, there needs to be an entity which is interested in establishing a federation of instances. That entity shall become the manager and administrator of the federation. Second, there need to be instances which are interested in joining the federation. Once the different entities are in place, they can create a consortium which shall collaborate within a federation.

The rules governing the federation consortium are out of scope of the current work and are left for the different entities who wish to collaborate to decide. A technical requirement for instances to join a federation is to support the APIs of the NIMBLE core services[3]. Once the governing rules and technical setup are in place, the federation manager can spin up the federation core services and provide the required configuration to the individual instances wishing to join. The individual instance will in turn register itself with the federation and the collaboration can begin. This process is somewhat analogous to a new company that wished to join an instance.

4. Cross Federation Query Execution

The Delegate service handles queries either by redirecting them to the specified instance or sending them to all instances and merging the responses. There are two types of queries running through a federation:

Distributed queries: The Delegate service executes such queries in all available instances by redirecting the query to the delegate service of each instance. When responses are retrieved from the respective instances, it merges them and returns a single response to the client. For example, the delegate service redirects keyword-based search queries to all instances, merges the responses, and returns all matching products to the client.

Direct queries: In cases an instance keeps the details of an individual resource, such as a product, the query is executed only in the respective instance. The identifier of the target instance would be available for the querying instance, as described below.

4.1. Cross Federation Identification of Resources

Since each instance has an isolated storage for resources such as company information or product details, it is not possible to uniquely identify a given resource across the federation. Therefore, an additional identifier, called instance identifier, is attached to resources indicating their owner instance. As described below, while instance identifiers are persisted along with the resources themselves for some resources such as companies; there are cases where they are attached in runtime.

Persistent Ids: Considering the way federation functionalities are implemented, there is a need to persist a resource from a source instance in a target instance. This creates a data integrity problem as the identifier of the resource might not be unique across the federation. Therefore, resources (specifically company and transaction list) utilized in such cases are expanded with the instance identifier.

Runtime Ids: When the query is distributed to all instances, the client delegate service assigns an instance identifier to each response retrieved from other instances. This is the case for the distributed keyword-based product search queries. The client delegate attaches the instance identifier to each result item so that it can be used for subsequent queries e.g. to retrieve the complete details from the respective instance.

5. Cross Federation Activity Flows

Overall the NIMBLE platform is composed of a set of microservices with dedicated roles. The identity service mainly performs authentication and authorization to validate users' credentials and their rights to perform certain activities. The catalogue service deals with the persistence and querying of product information. The business process service provides B2B collaboration management capabilities.

5.1. Product/Company Search

The primary feature to be federated is the product or company search. That is a pre-requisite for proceeding with the subsequent B2B activities. For keyword or product classification-based search, the client delegate sends the search query to the delegate services of all instances. Each delegate upon receiving the search request invokes the query on its local search component and sends back the results to the initiating delegate. The delegate at the initiating instance combines all the results received, locally or remotely, adds the instance identifiers referring to the owner instances and returns the combined set of results to the client. Search results retrieve the basic information concerning the relevant product or service. The user can drill down to obtain more information on specific products or services, by clicking on the item of interest in the front-end. For local products the front-end turns to the catalogue service which returns the desired information. For remote products there is a need to query the remote catalogue service. To realize this capability the federation aspects come into play. The call which in the local case targets the local catalogue service, turns instead to the local delegate. The delegate, using the mechanisms described above of the core federation services, locates and contacts the remote delegate service on the corresponding instance. The remote delegate invokes the call on its local catalogue service and the response is sent back to the originating delegate. The originating delegate in turn sends the responses back to the local front-end to be presented to the interested end-user. The front-end discovers the local delegate using the local discovery service and directs the search call to the local delegate. The local delegate shall turn to the local search component and invoke the corresponding query on it. In parallel, the delegate shall query the federation core services to obtain a list of all available delegates of other instances in the federation. Each delegate receiving the search request shall invoke the query on its local search component and shall send back the results to the initiating delegate. The delegate at the initiating instance shall combine all the results received, locally or remotely, shall add an identified as to the originating instance and send back the combined set of results to the local front-end. Up-to-date company or user information is needed frequently e.g. to retrieve default delivery address, trading terms, associated users of a company. The delegate uses only direct query execution as the requested resource is located in exactly one instance.

5.2. B2B Processes

To achieve cross instances business processes, we require synchronization of the relevant state across the participating instances. The business process service employs an engine to keep track of the B2B data exchange flows between trading companies. In this sense, there were two main options for keeping the remote instance synchronized with the source instance concerning the activities: 1) using a single business process engine in one of the instances or 2) synchronizing engines on both instances.

In order not to face synchronization issues between instances with respect to engines and data models used to represent business processes, we decided to keep the information on the seller instance in case the trading companies reside in different instances. The engine of the seller instance is used to maintain the only state of a business process involving trading companies from different instances. The counterpart (i.e. the buyer) will interact with the business process engine remotely via the delegate service. Within a single instance at each step of the business process there is a party which sends a message, and the other party which is waiting to get a message. The federated case will be handled in a similar manner, having the seller side operate regularly (locally), while the buyer side will operate remotely using the previously described delegate mechanism.

After initiating the process in a particular business process engine, triggering events on it follows the same path as search requests: end-users local to the business process call it directly, whereas remote end-users call their delegate, which forwards their calls to the delegate local to the business process, which in turn invokes the business process. Thus, the entity in the instance which is using a local engine will invoke the API directly, while the remote instance will invoke the API remotely using the delegate processes on both instances. Entities polling for information will have to poll all instances in the federation to discover the updated information concerning their active business processes across all instances in the federation. The NIMBLE dashboard for business processes is populated in that manner. When populating the dashboard with the updated state of all business processes, that call shall be distributed to all members of the federation, using the delegate service, to obtain complete and updated information.

6. Acknowledgements

This research has been funded by the European Commission within the H2020 project NIMBLE (Collaborative Network for Industry, Manufacturing, Business and Logistics in Europe), No. 723810, for the period between 01 October 2016 - 31 March 2020

7. References

[1] ATRASH A, ABEL M., “Supporting organizational learning with collaborative annotation”, 16th KMIS, 2014, p. 237-244
[2] Heikkurinen, o. et. al., “Federated ICT for global supply chains”, IFIP, 2013, p. 1268-1275
[3] Innerbichler, J., Gonul, S., Damjanovic-Behrendt, V., Mandler, B., & Strohmeier, F. (2017, June). Nimble collaborative platform: Microservice architectural approach to federated iot. In 2017 Global Internet of Things Summit (GIoTS) (pp. 1-6). IEEE