Enterprise SOA and MDM: Multiple Service Providers Framework

Applies to:

SOA, MDM, Visual Composer, enterprise SOA, Composition Environment

Summary

SOA is the organizational freedom of choice. Instead of having to purchase a complete software package for providing a specific, unique solution, organizations now have that functionality decomposed to a single, stand-alone execution unit, available for consumption as a web service, easily integrated into your business process.

This article will suggest a generic framework leveraging multiple service providers and explain how SDS (Strategic Data Services) implemented this framework, together with its data quality partners, using MDM and SAP's Composition Environment.

Author(s): Yossi Ben Harush

Company: SAP

Created on: 20 June 2007

Author Bio

Yossi Ben Harush is a configuration engineer for MDM SDS (Stratigic Data Services), a team providing data quality services. Among his dueited he is building composite application around SAP MDM, using SAP's composition environemnt.

Table of Contents

SOA Evolution

SOA is maturing. The enterprise understands the strength and flexibility of implementing SOA, ISV's realize that in order to remain relevant they have to develop web-service interfaces to the functionality they offer and the platform companies provide the composition environment necessary to support the architecture.

SAP, with its NetWeaver offering, is well ahead of its competitors in providing a robust, production-quality SOA infrastructure. Visual Composer enables fast development of composite application, easily consuming web services. Guided procedures is the standard tool for modeling business processing, with strong functionality to supports roles. SAP MDM is the fundamental building block of any SOA implementation, enabling a central point of web service consumption.

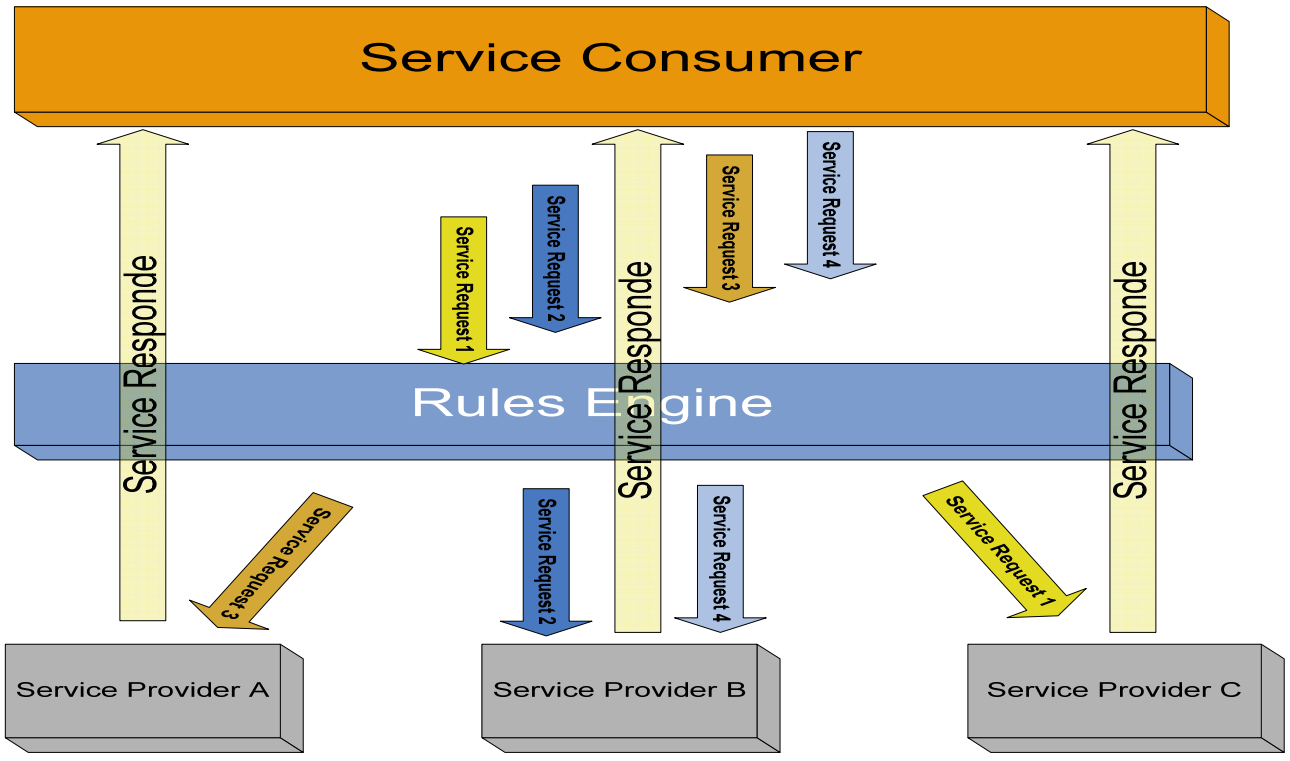
SAP also has a strong partner ecosystem that allows its customers a true freedom of choice.

I will suggest here a general framework for a smart consumption of web services, leveraging multiple service providers. This framework is relevant in an environment were more than one ISV can provide response to a single, well defined request. In that case, a decision has to be made on which provider will be the processor of each specific request. I will provide a more detailed description of how this framework was implemented to provide data quality services for SAP MDM.

Multiple Service Providers Framework

The framework suggested here contains 3 basic components:

- Service Consumer - A Backend system (SAP ERP, SAP MDM etc.) requesting services.
- Rules Engine – A flexible business logic layer, containing rules for deciding which service provider to consume.
- Multiple Service Providers – 2 or more web services from different service providers capable of answering the service request.



Service Consumer

The service consumer can be any backend system requiring a specific service. Such systems might be ERP, CRM , SRM or MDM. Using SAP MDM as the service consumer makes a lot of sense because as MDM can act as a central access point to all available services – you do not need to consume the server from each backend system – MDM can do that on behalf of all other systems.

The request for service has to be atomic in the sense that the functionality required by the consumer is well defined and is independent of any other part in the business process. All information required for the request should be available and well defined so that the service providers can get all necessary information in order to be able to process the request and provide a proper response.

The example we will use is an MDM scenario (Customer Address Validation) that was presented by SDS in the MDM-CDI summit during March 2007.

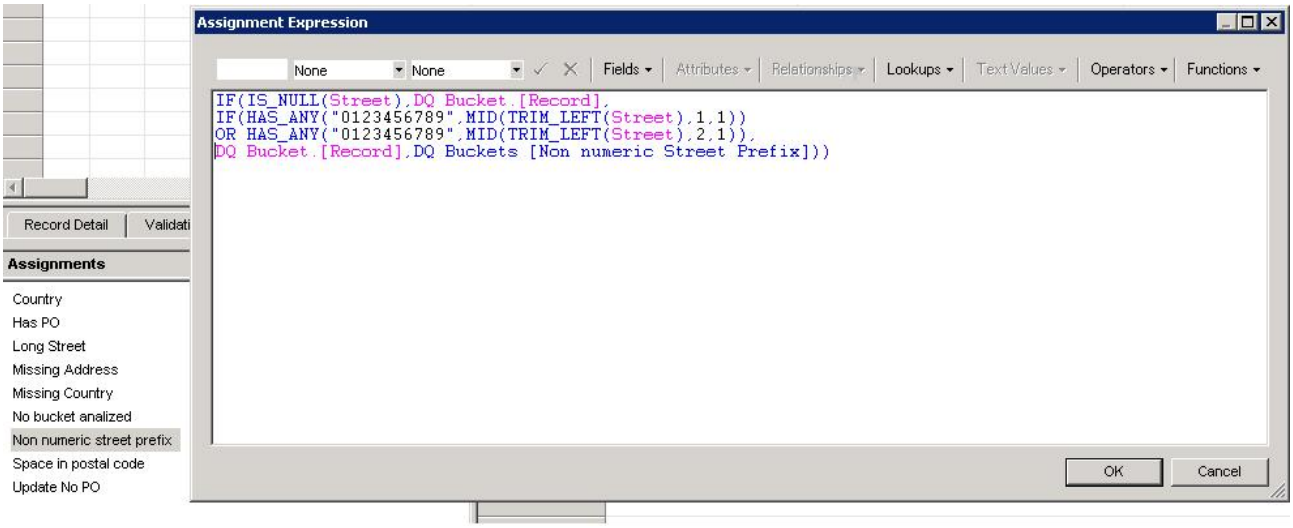
The service consumer in our example is SAP MDM. MDM holds master data information coming from multiple backend systems – in our case, since we process customer data, possible systems are ERP, CRM, billing systems etc.

MDM will request a well defined service – address validation. Customer data will be exported from MDM (using syndication server) and through MDM Enrichment Controller will be passed to a service provider. The format of the request is also well defined – Customer name and address information (Street, City, Country etc.) defined by an XSD file. The Address Validation service request can be handled by several service providers (4 in our example) – all of them offer address validation services.

Rules Engine

The novel part of the suggested framework is a business logic layer that resides between the service consumer and the services providers. Its purpose is to decide, according to predefined rules, from which service provider to request the service. In such a scenario, each request is first sent to the Rules engine. The rules engine applies pre-determined logic on each of the records in the request and decides which of the service providers will process this request.

We will use our address validation service – Let's assume we are sending 100 customer records to an address validation service. All records will go through the Rules Engine for classification. Rules will be applied on each record to decide to which service provider the record should be sent. In our implementation, we used SAP MDM's assignment rules functionality to mark each record with a "bucket code" – all records that met the same condition belong to the same bucket. Rules that might be applied in the case of address validation: "address is in the United States and the street address does not start with a digit", "Address is in the U.S. and zip code is not 5 or 9 digits", "country code is missing". The idea behind these rules is that for each rule, there is a service provider that can handle those records better than others. For example, some address validation services cannot validate an address without getting the country from the service consumer.



Multiple Service Providers

The basic assumption of the proposed framework is that a specific, well defined service can be requested from multiple service providers.

For ISV's, it makes sense to publish there software as web services. Their customer base can now grow to any service-consuming client. It allows them to highlight their strength and creates a more competitive market where the customer doesn't have to decide on one specific solution but can get the best possible service for each request he is sending, according to rules he has created. It is likely to create competition between service providers trying to be the chosen vendor for each request, improving the overall quality of solutions, providing higher value for customers implementing such framework.

In our example, SDS teamed with four ISVs, all of them offering the same service – address validation. The exact format of the request and of the response were agreed with the ISV's, so that any Address Validation request sent from MDM can be processed by all 4 vendors.

Gluing together the 3 components mentioned above requires a strong SOA foundation. At the core of the framework lies SAP MDM. Since MDM can gather information from multiple backend systems, it make the whole process more sensible. You do not need to integrate each R3 or CRM instance you have into the composite – you do it only once, with MDM. In our example, the backend systems can export data to be imported into MDM. MDM will then consume the service and syndicate all processed data back to all backend systems. Implementing this scenario is much simpler than trying to have each backend system as service consumer.

Another advantage MDM has as a single-point service consumer is the Enrichment Controller Framework – it defines a generic method of data transport from SAP MDM to 3rd party software providers. For the example we used, 4 different adapters (instances of the enrichment controller) were written together with four different data quality vendors to provide, among other, address validation services.

MDM also has a built-in rules mechanism that can be used to implement the business logic layer. Rules such as the ones that were mentioned earlier are very easy to compose and apply in MDM.

Composition environment

Another important part of implementing such a framework is Visual Composer. Consuming web services in Visual Composer is a matter of minutes. VC also has its' own rules engine so the Rule Engine layer can be implemented in VC as well. In order for VC to be able to integrate the 3 layers, the following steps need to be taken:

- The backend systems need to expose the data that needs to be processed by the service.
- Rules have to be defined in VC in order to decide to which service provider the request has to be routed.
- The service providers need to provide web service interface to their functionality.

Once all 3 steps have been taken, a composite can be created.

It is also possible to use both MDM and VC together to build the composite, depending on the specific requirements.

Related Content

Please include at least three references to SDN documents or web pages.

- Enterprise SOA in a Nutshell – A good starting point for SOA.
- MDM Data Enrichment – Information related to MDM Enrichment Controller.
- Consuming Web Services Using Visual composer – Learn how easily you can implement the suggested framework using Visual Composer.

Copyright