James Wood

January 4, 2013 | 5 minute read

Navigating the BOPF: Part 1 – Getting Started

15 65 61,045

Last year, I began working on a project which was rolling out the new SAP *Environment Health and Safety Management* (EHSM) module. This module utilizes some of the more cutting edge technologies in the ABAP space: Floorplan Manager, Web Dynpro ABAP, Adobe Interactive Forms, and so on. In addition to these standard technologies, EHSM also utilizes a technology framework that I had not encountered previously: the *Business Object Processing Framework* (or BOPF).

Whenever I started investigating the BOPF, the first place I went to look was naturally right here at the SDN. However, between SDN and Google, I found very little information to go on when it comes to working with the BOPF. Indeed, about the only useful documentation I found was an enhancement guide entitled *BOPF Enhancement Workbench*. What I was really looking for though was an in-depth description of the architecture of the BOPF, its API, and most importantly, some examples demonstrating its usage. Short of that, I was left to muddle my way through much of the SAP standard code until I began to understand how the different pieces fit together.

After working with the technology for the better part of a year, I thought I would launch a blog series documenting my findings for others who share a similar plight. This includes other EHSM developers as well as developers working in other

new dimension modules such as SAP *Transportation Management* (TM), etc. I hope you will find it useful.

What is the BOPF?

As the name suggests, the BOPF provides a framework for working with business objects (BOs). This framework provides tools and services which span the entire BO lifecycle:

- Design Time
  - At design time, BOs are modeled using the BOPF Workbench tool (Transaction /BOBF/CONF_UI). This tool makes it possible to model a BO's nodes/attributes, behaviors, associations, and so on. If you're comfortable with OOP concepts, then this will seem vaguely familiar to modeling classes in the Class Builder tool. (Note: ~~So far, it seems that this tool is locked down for customer use. This implies that we cannot create new BOs of our own...yet.~~ As per Thea Hillenbrand's comments below, the BOPF has been opened up for general customer use. This happened with SAP Business Suite EHP5 SP11 and SAP Business Suite EHP6 SP05. The related note is 1760610. Thanks Thea!)
  - Behind the scenes, the BO metadata is stored in such a way that it can be introspected and leveraged by runtime APIs.
  - Customers can enhance existing BOs using the BOPF Enhancement Workbench tool (Transaction /BOBF/CUST_UI). Here, we have the option of defining new nodes and attributes, defining additional behaviors, and so on. We'll see how this works up close in an upcoming blog entry.
- Runtime
  - At runtime, BOs are instantiated and controlled via a standard API defined using ABAP Objects classes.
  - Transactions are managed by a central transaction manager class.
  - Service-level interactions are brokered via a standard service manager class.
  - To some extent, much of this will feel similar to ABAP Object Services. However, as you'll soon see, the BOPF affords us a lot more power.

The figure below illustrates how these different pieces fit together within an application. As you can see, the BOPF architecture utilizes a layered approach:

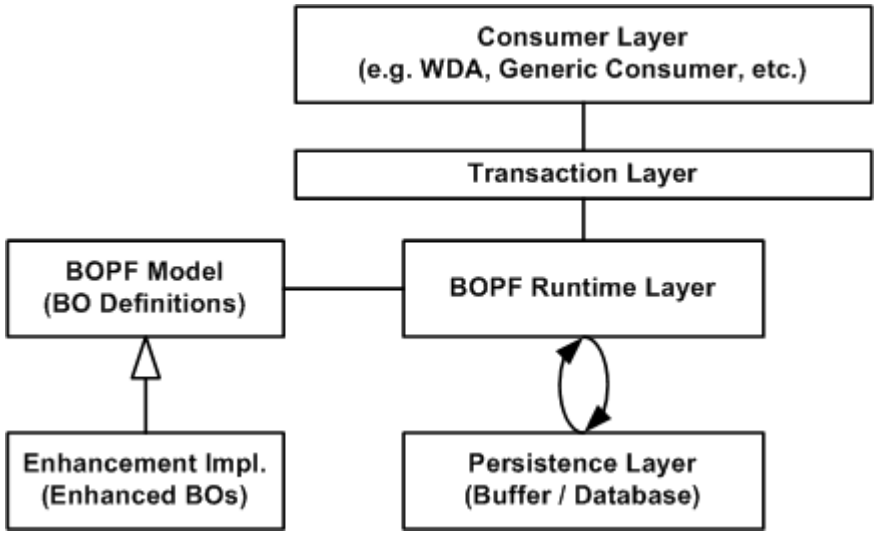
- Consumer Layer
  - At the consumer layer, we can utilize the BOPF API methods to create new BOs, search for existing BOs, update selected BOs, and so on.

- Frequently, BOPF BOs will be consumed by UI applications such as WDA applications, etc.
- Of course, that's not to say that generic consumers cannot get in on the fun as well.

- Transaction Layer
  - Interactions with BOs within the BOPF are brokered through a centralized transaction layer which handles low-level transaction handling details such as object locking, etc.
  - From the perspective of the consumer layer, interactions with the transaction layer consist of little more than a handful of intuitive API calls.

- BOPF Runtime Layer
  - The core of the BOPF functionality lies within the BOPF runtime. This layer contains all of the functionality required to instantiate BOs, trigger their functionality, and so on.
  -
  - As you can see in the figure, the BOPF runtime utilizes the BOPF model definitions created at design time as metadata for instantiating BO instances, navigating BO associations, etc.
  -

- Persistence Layer
  - One of the nice things about the BOPF is that it is rather flexible at the persistence layer. Though the end goal is normally to store BO data within the database, the framework also supports data buffering via shared memory as well as the definition of transient nodes and attributes that are loaded on demand.



Though the BOPF shares certain similarities to previous business object models (e.g. business objects defined in Transaction SWO1, GENIL, and BOL), it is quite a bit more evolved than any prior business object model defined by SAP. This will become obvious as we delve into specific topics in the upcoming blog entries.

Why do we need the BOPF?

Whenever a new development framework comes out, it is only natural for developers to wonder if the framework is truly needed. Though I will not endeavor to sell anyone on the merits of the BOPF within this blog series, I think it is useful to compare and contrast the scope of a development project with and without the BOPF. Then, you can decide for yourself if the BOPF provides value.

To put this in perspective, let's imagine that we're tasked with developing a new module around some custom-defined business entity. A minimal bill of materials in terms of development objects for this module are as follows:

- The entity data will be stored within a series of ABAP Dictionary tables. If desired, we can use ABAP Object Services to create an ORM wrapper around these table accesses.
- In order to synchronize access to the entity data, we'll need to define one or more lock objects. Plus, we'll need a mechanism to ensure that the lock objects are used to control access to the data.
- To secure access to the entity data, we must create one or more authorization objects so that we can define granular authorization control.

Now, in an an ideal world, we would attempt to encapsulate access to the entity data by creating a series of ABAP Object classes (e.g. entity classes and the like). These classes would offer basic CRUD (Create, Remove, Update, and Display) operations that provide users with a one-stop shop for accessing and updating the entity data.

With these basic components in place, we can then build the core application functionality. Here, we'll find a number of producers/consumers of the entity data:

- UI applications based on WDA, BSP, or even classic Dynpro technology.
- BI extractors used to export the data for reporting purposes
- SAP Business Workflow processes
- Custom accessor modules used to supply data to home grown reports (e.g. ALV), Adobe Interactive Forms, and so on
- Interface and conversion programs (or, in SOA initiatives, Web service wrappers)
- Others as needed

Overall, these kinds of development objects are pretty standard fare for the average ABAP developer. However, within the context of your typical SAP project, these tasks are often distributed across a large development team. Here, it can be difficult to enforce best practices and ensure that each developer accesses the entity data properly. Frequently, this is as a result of developers not understanding how to access the data via a custom API (provided there is one). In short, there's no overarching object model which ensures that business objects are accessed consistently.

It is here that the BOPF shines. Within the BOPF, everything has its place. Business data is modeled consistently in BO nodes and attributes. Behaviors are defined as actions. Validations are performed automatically via validation modules. Triggers can be defined using determinations. The relationships between BOs is defined statically via associations. Once a developer becomes comfortable with the framework, the BOPF takes all of the guessing out of business object development. The BO encapsulates all of the functionality and provides consistent access to all of the producers/consumers outlined above.

In time, this sort of consistency can give rise to additional frameworks which sit on top of the BOPF. An example of this is the *Floorplan Manager BOPF Integration* (FBI) framework which simplifies the way that FPM feeder classes access BOPF nodes. Much of this will become clearer as we move on. For now, suffice it so say that the BOPF provides us with a tremendous springboard for developing business objects.

Next Steps

Now that you have a basic feel for what the BOPF is all about, we'll move on to some more specific topics of interest. In my next blog, I'll tackle BOs from a design perspective.

Assigned Tags

ABAP Development

abap

bopf

floorplan manager

Web Dynpro

web dynpro abap

Similar Blog Posts

Related Questions

Where the integration (link) of Bopf_ewb and Fpm done

By Former Member Nov 25, 2015

BOPF-Business Object

By mustafa cengiz Aug 17, 2020

How to call another BOPF from action of one BOPF ?

By Surya Sarathi Basu Dec 26, 2018

15 Comments

You must be Logged on to comment or reply to a post.

Former Member
January 26, 2013 at 12:53 pm

good info

Like 0 | Share

Felipe Hernandez Plazas
January 30, 2013 at 2:42 pm

Hi James,

excellent post!

Like 1 | Share

Aliaksandr Shchurko
May 2, 2013 at 12:58 pm

Thanks.

Like 0 | Share

Former Member
June 4, 2013 at 4:57 pm

Hi James,

Congratulations on this excellent series on BOPF. We feel a bit ashamed not having delivered these informations by our own.

BOPF was an internal tool to ease and govern our application development. Last year we saw an increasing interest from partners and customers and we decided to open it also for customer usage. Not the /BOPF/CONF_UI transaction - good enough for internal usage - but the enriched enhancement workbench which we renamed to Business Object Builder (transaction BOB). This transaction can be used to create new BOs and enhance existing ones.

This happend with SAP Business Suite EHP5 SP11 and SAP Business Suite EHP6 SP05. The related note is 1760610.

Regards

Thea

Like 2 | Share

Fred Verheul

October 12, 2013 at 6:38 pm

Hi James,

Always a bit slow, I'm just starting to explore the BOPF via your blog series. Thanks for taking the time to compile this, excellent job!

Cheers, Fred

Like 0 | Share

Tudor Riscutia

March 5, 2014 at 8:00 pm

Hello James,

Excellent introduction into BOPF, I've shared this in my team and recommended it to anyone curious about this topic. Please, be so kind and continue writing articles about BOPF.

Greetings from Romania!

Tudor

Like 1 | Share

Rohit Mahajan
August 18, 2014 at 11:57 am

very good document,,,

Like 0 | Share

Jeena George
April 20, 2015 at 10:20 am

Hello James,

Thanks a lot for the detailed document.

Cheers,

Jeena George

Like 0 | Share

david hurtado
October 20, 2016 at 7:08 pm

learn is good

Like 0 | Share

Avinash D M
August 21, 2017 at 7:41 am

Very good information , Thanks

Like 0 | Share

Srinivasarao Pydisetti
May 29, 2018 at 6:53 am

Hi James,

Thanks for your inputs. I appreciate!!.

Thanks,

Srinivas.

Like 0 | Share

Pavan Golesar
April 2, 2019 at 8:01 pm

Hi,

nice blog!! 🙂

Br,

Pavan Golesar

Like 0 | Share

Zhang Chao
September 10, 2019 at 2:40 am

Nice blog!

But a little mistake, the means of CRUD should be "Create, Retrieve, Update, Delete".

Like 0 | Share

Channa Reddy TR

October 11, 2020 at 6:33 am

Hello James Wood,

Excellent blog, Thanks a lot for sharing.

Best Regards,

Channa.

Like 0 | Share

Prasenjit Bist

June 11, 2022 at 5:06 am

In 2022 I am working on EHSM implementation for a North American customer, and again reading through your BOPF material. The very first SAP press book I bought was also authored by you (ABAP Objects) and the very first BOPF project (SolMan implementation), I did for a German customer in 2017-2018 was also after reading your blogs 🙂

These blogs are worth a book and should be preserved in SAP community as the definitive tutorial given there are no credible blogs like this and there are no SAP documentations worth mentioning.

Thanks James Wood

Regards,

Prasenjit

Like 0 | Share

Find us on

Privacy	Terms of Use
Legal Disclosure	Copyright
Trademark	Cookie Preferences
Newsletter	Support