



## PPO™ Integration and Interoperability

This document provides a high-level overview of the architecture of PPO™ from an integration and interoperability point of view and describes options available to solution integrators and architects.

### Logical Architecture

PPO™ uses an n-tier architecture which basically means that there are several logical layers or components that make up the system. The three main layers are the data layer, the application layer and the presentation layer.

Of these layers, the application layer is the most important from an integration point of view. All functionality within PPO™ can be accessed through this layer via an API (Application Programming Interface). Three other components of PPO™ that are important from an integration point of view are the Event Management System, the Mail Management System and the Rule Processor.

Each of these components is discussed in more detail below.

### Application Programming Interface

The core PPO™ application library, which was developed in Microsoft .Net, exposes over 600 methods that can be used to read or update information in PPO™. Through this API, all functionality within PPO™ is available to developers that want to integrate PPO™ with other applications, automate certain tasks or provide an alternative front-end such as a corporate web portal.

### Event Management System

Whenever an event occurs within PPO™, such as when an item is added, updated or deleted, the PPO™ application tier places a message on an MSMQ (Microsoft Message Queuing) queue. The Event Management System monitors this queue and determines whether an event handler has been set up to deal with the particular type of event. If an appropriate handler is found, it calls it and passes it the information pertaining to that event. Currently, there are three different types of event handlers that can be set up, namely:

*Application event handler:* This type of handler calls a command line application and passes it the information pertaining to the event as an argument in XML format. The command line application is usually a custom developed application that does the actual work required for the particular integration scenario.

*Script event handler:* This type of handler calls a script, which is basically an uncompiled piece of .Net code, compiles it on demand and executes it, passing it the event information as a parameter. This type of handler is useful if frequent changes are required to the business rules associated with the particular integration scenario.

*Mail event handler:* This type of handler sends an e-mail based on the event. The information in the e-mail as well as the intended recipients is determined by the business rules implemented in the event handler.



## Mail Management System

The mail management system within PPO™ contains a feature by which it can be instructed to monitor a specific mailbox account. When mail gets delivered to that mailbox, it places that message on the MSMQ queue, where it will then be handled by the Event Management System as discussed in the previous section. This is a useful feature in that it can be used for inbound integration (i.e. another system that wants to update something on PPO™). It can do this by simply sending a mail message to a specific mailbox.

## Rule Processor

PPO™ includes a rule processing component, which make it easy to set up event handling based on complex business rules, using a simple XML based configuration file. This can for example be used to notify the person that a task is assigned to a day before the planned end date for those tasks which have not been marked as completed.

## Practical Examples

The following examples, illustrate some integration scenarios, and how it can be accomplished within the PPO™ architecture. Many of these examples come from actual integration done at our clients.

### Example 1: Integration with HEAT (a call logging system)

The client uses HEAT as a call logging system. Whenever a call is created on HEAT, a corresponding issue must be created on PPO™ and assigned to a 3<sup>rd</sup> party contractor to address. If the call is updated on HEAT the issue on PPO™ should also be updated and vice versa.

Since both HEAT and PPO™ support e-mail based notifications and updates, the integration was accomplished by HEAT sending an e-mail to a PPO™ mail box as soon as a call is logged or modified. An event handler was then set-up on PPO™ which creates or updates the issue on PPO™ and assigns it to the relevant person. Similarly, if the issue is updated, an event handler sends an e-mail to the HEAT system in the prescribed format, resulting in it being updated on HEAT.

This type of integration can be used in a wide variety of scenarios, including those where the systems to be integrated are on disparate networks, or not always available on-line.

### Example 2: Integration with SAP

The client uses SAP PS and wants to update project cash-flow forecast in SAP PS from PPO™.

A web service was developed in SAP XI, which is called by an event handler that was set up on PPO™. Whenever project costs are updated, the event handler calls the web service, which in turn updates SAP PS.



### **Example 3: Integration with a custom web-application**

The client has a simple, custom developed web application which shows key milestones per project and allows users to update the % complete of the milestones. The requirement was for the web application to show the projects and milestones from PPO™ and to update the relevant tasks on PPO™ in real-time.

The web application simply calls a web-service synchronously on PPO™ to get the list of projects and milestones and displays it to the user. If the user updates the % complete, another web service on PPO™ is called, which updates the task with the % complete.

### **Example 4: Providing information to a data warehouse**

The client wanted project risk information to be fed to its data warehouse on a daily basis.

A scheduled task was created, which calls a script developed in C# that pulls the required risk information from PPO™ via the API. It then builds a text file from this and FTP's the file to the warehouse staging area.

**For assistance and more information relating to this factsheet please contact us @ [support@postvision.co.za](mailto:support@postvision.co.za).**

**For a detailed description of PPO™ functionality, please refer to the Product Brochure.**

### **Post Vision Technology (Pty) Ltd**

**Tel:** +27 (12) 348 2366

**Fax:** +27 (12) 348 8382

**Email:** [info@postvision.co.za](mailto:info@postvision.co.za)

**Web:** [www.postvision.co.za](http://www.postvision.co.za)

**Physical Address:** Ground Floor, Building 3, Ashlea Gardens Office Park  
Cnr. Garsfontein & Matroosberg Roads, Ashlea Gardens, Pretoria

**Postal Address:** PO Box 75760, Lynnwood Ridge, 0040, Pretoria, South Africa