

AVOKA SMARTFORM FACTORY – ARCHITECTURAL PRINCIPLES

A FACTORY-ORIENTED APPROACH TO SMARTFORM DEVELOPMENT

Please refer to the Avoka white paper, “Enabling Digital Self-Service” for background on the reasons why more complex paper-based transactions have not yet made it to the digital age, and why a factory-oriented approach to developing SmartForms is the solution to this problem.

These findings from the white paper are summarized in the following table. We also refer to the specific module or feature of SmartForm Factory that implements each requirement.

Requirement	SmartForm Factory Feature or Module
The overall system should be managed by IT, but forms should be designed and managed by the business.	SmartForm Factory
You need a tool that allows you to build sophisticated and useable forms easily, quickly and reliably.	SmartForm Composer
Optimise the forms you build to render “naturally” on whatever device your customers choose.	Design once run everywhere
Your system must replicate many of the attributes of paper forms	SmartForm “Apps”
Forms must integrate with your existing web site	Finder, Web Plugin, Portal, Adobe WEM
Your system must recognize that different users have different usage models	Self-Service Portal vs Self-Service App vs Field Worker
Forms must support integration with your backend systems	Integration Agent, Adobe Digital Enterprise Platform
Scalability, Manageability, and other –abilities	SmartForm Manager

These modules are also shown in the following diagram:

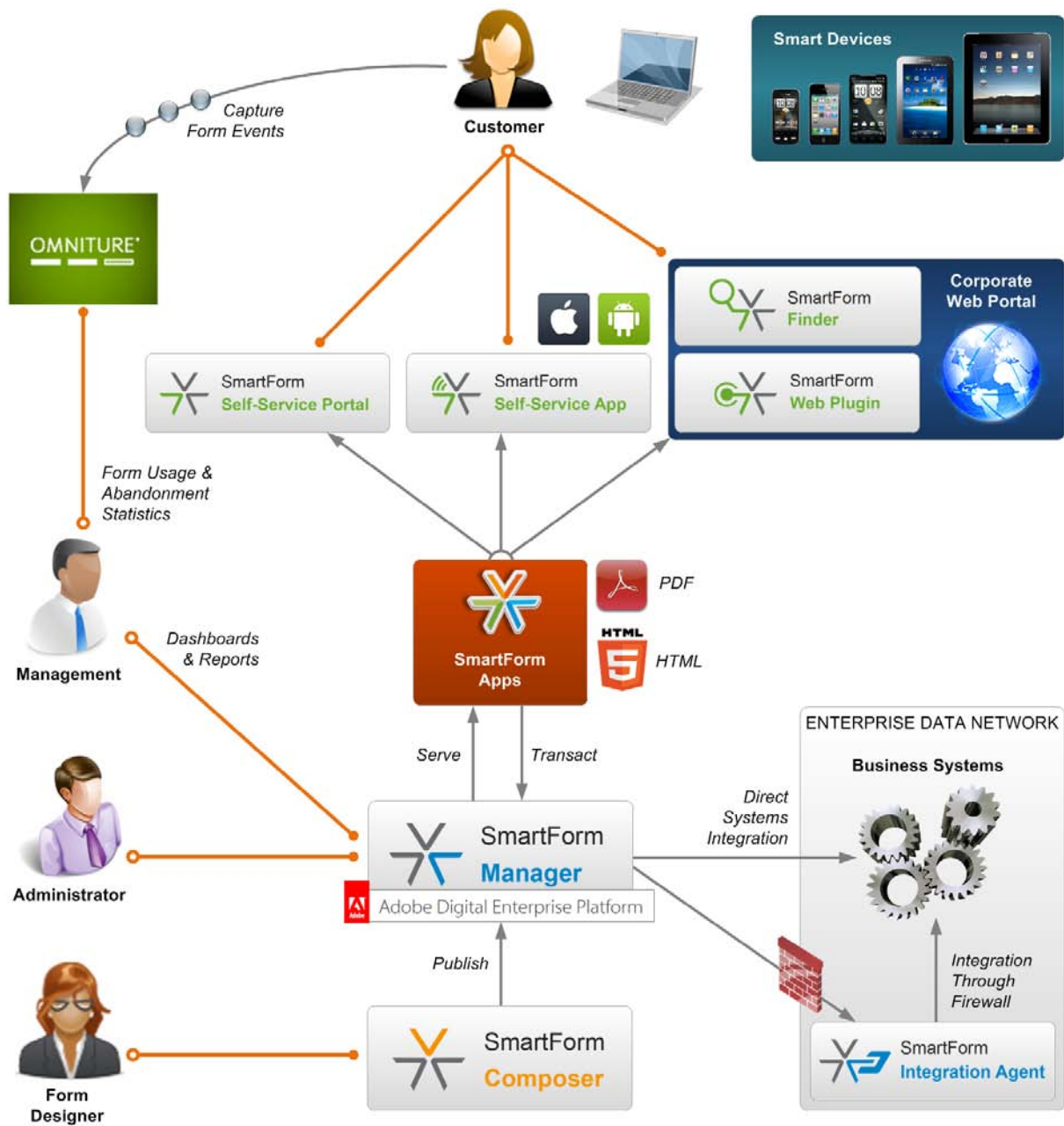


Figure 1 – High Level Overview of SmartForm Factory Modules

The rest of this paper describes the various modules and features, and how they work together to enable a comprehensive solution to digital self-service.

SMARTFORM FACTORY

SmartForm Factory is a complete offering to bring complex paper transactions into the digital age. SmartForm Factory consists of a number of complimentary and cooperating modules that together change the way in

which organizations can create, publish, consume and integrate SmartForms. In turn, this allows the organization's customers to discover SmartForm Apps, complete them online or offline, and transact with the organization.

SmartForm Factory enables a paradigm shift in the way that SmartForm Applications are created and managed:

- The IT Department installs and maintains the SmartForm Factory platform.
- The Business creates, publishes and manages the SmartForm Apps that are created and hosted on the platform.

This paradigm shift allows SmartForm applications to be designed, developed and deployed quickly and easily, without being restricted to IT development timeframes and constraints.

SMARTFORM COMPOSER

At the beginning of the production line that constitutes the SmartForm Factory, there must be a way to quickly and easily create sophisticated SmartForm Apps. Without this capability to produce the "raw materials", the rest of the production line breaks down.

SmartForm Composer has been built by form developers for form designers. Avoka's SmartForm experts have collectively built thousands of digital forms ranging from the simple to the insanely complex, and we have encapsulated everything we've learned into Composer. This allows you to build a form in just a few hours that would have taken one of our experienced form designers days or even weeks to build.

A full description of all the features of Composer is well beyond the scope of this short paper, but a few core features include:

- **Caters for Left and Right brained users.** Designing forms is a surprisingly difficult process, requiring both design skills (right brain) and programming skills (left brain). These complementary but opposing skills are rarely found in the same individual, with most people dominant in either left or right brain. Composer actually allows you to build a form without needing deep skills in either area: Your forms always look good, because they conform to the organization style guide; and they work well, because Composer encapsulates most of the programming required to make forms highly functional in drag and drop widgets.
- **Sophisticated style sheets.** Style sheets control the majority of the look and function of the forms you develop. This means that form developers can focus on building the forms, and let the style sheets ensure that their forms look and function according to the organizational style guide first time, every time. Style sheets can control almost everything about a form, from colours and fonts to whether a form is wizard-style or paginated vertically.
- **Extensive re-use.** Composer is built from the ground up, from a fully object-oriented shared widget hierarchy and user-definable blocks, to a fully functional organizational Data Dictionary to enable the best possible re-use across the entire organization.
- **Rule-oriented.** Many form-development environments require programming skills in order to add any kind of dynamic behaviour. Composer is rule-oriented rather than script oriented, so that non-programmers can easily create business logic in their forms.
- **Automatic layouts.** Rather than forcing you to laboriously lay out your forms, lining fields up, and remembering widths, heights and gutter settings, Composer uses layout managers and "hints" to lay out your forms perfectly.
- **Data Dictionary.** Composer includes a very powerful data dictionary that can be used to manage sophisticated data models. The dictionary can be used as a starting point to create forms or to ensure

the consistency and integrity of data across all forms. The data dictionary can also serve as a starting point for integration with backend systems meaning the information collected in forms always fits with the databases of core business systems

- **Enterprise grade.** Composer is not a toy. It is designed specifically for large enterprises, with sophisticated requirements not only for the forms that are developed, but also of the process used to develop the forms.

DESIGN ONCE RUN ANYWHERE

The modern computing world has already progressed way beyond the traditional desktop – users now interact from a variety of different types of computing devices, including desktop, smart-phone and tablet, each with different capabilities, technologies, screen sizes, and modes of operation.

You need to be able to design a form once, and have it display and function correctly on whatever device the user chooses, whenever they choose to use it. Importantly, it’s not just sufficient to display the same form on a different device, the display and function of that form must be optimized to provide a natural “fit” for that specific device and it’s capabilities.

Some examples of making the form “fit” the device are explained below.

- **One Size Doesn’t Fit All.** PDF forms are traditionally the “gold-standard” for more complex forms in many environments. However, PDF forms are designed for standard paper sizes such as A4 or Letter, often with multiple fields on the same line. An A4 sized form is going to be incredibly difficult to interact with using a phone, requiring the user to scroll left and right as well as up and down. A much better approach is to re-adjust the layout of the form to have all the fields flow straight down the screen, and also to break the form up into smaller sections that fit a smaller screen.

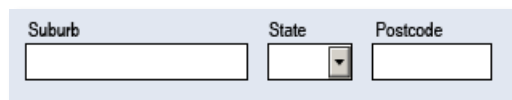


Figure 2 - A few simple fields laid out horizontally in a PDF Form



Figure 3 - The same fields should be automatically laid out vertically on a narrow smart phone

- **Device, Operating System, Browser + Software Compatability.** Technologies differ between devices. Adobe Reader is available on most desktop devices, and provides an optimum experience for filling in forms on these devices. However, there are alternate PDF readers on several platforms that do not support sophisticated PDF Form technologies. For customers with these environments, we need to be able to display the same form using different technologies, such as HTML5. Conversely, HTML5 may not always be appropriate, since many users still have older browsers that do not support HTML5, so older versions of HTML may be necessary.

- **Layout Look and Feel.** Tablet and smart-phone devices generally have a different look-and-feel to traditional paper or web-based forms. Ideally the form should adjust itself to look natural on whatever device they are being displayed on.



Figure 4 - A traditional PDF form may include a page header and a styled section heading

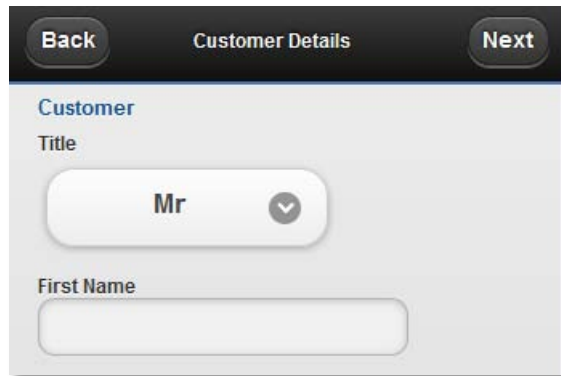


Figure 5 - The same section heading would be styled quite differently on a small smart-phone screen

- **Support Different Interaction Models (Touch/Mouse).** Tablet and smart-phone devices are designed for touch rather than a mouse. This has several implications for the way the forms work. For example, a traditional drop-down list is difficult to interact with using a finger, and is generally replaced with a touch-friendly scrollable pop-up list.

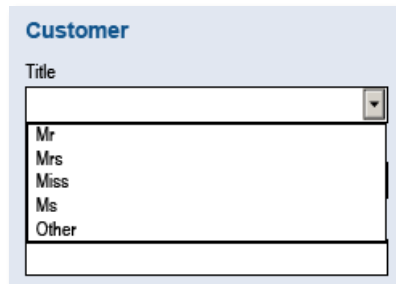


Figure 6 - A conventional Drop-down list in a PDF or desktop HTML form

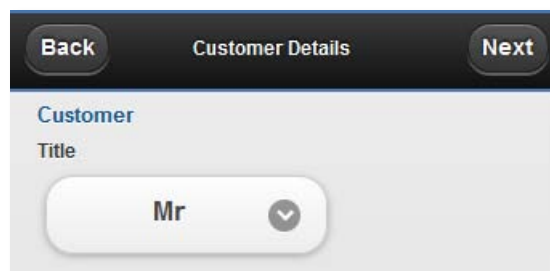


Figure 7 - The same dropdown list is displayed with a much larger area suitable for a finger



Figure 8 - And it pops up a finger-scrollable list box which touched

- Navigation Needs to be Different.** In a traditional page-oriented form, the user can easily scroll up and down a long form. However, on a phone or tablet, an easier way of navigating a large form is preferable, usually some sort of wizard-style interaction.

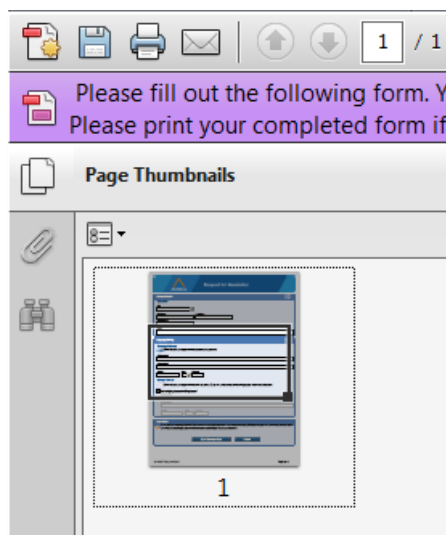


Figure 9 - A traditional PDF or HTML form can be scrolled, as illustrated in this thumbnail view

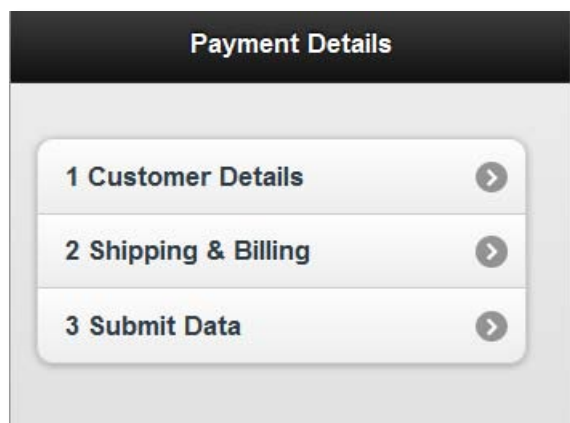


Figure 10 - The same form on a small mobile device would need to provide an optimized navigation paradigm

- **Error Handling Needs to be Different.** In a page-oriented form, you can easily display all the errors at the end of the form. On a small mobile device, errors need to be displayed and corrected as close to the point of data-entry as possible.

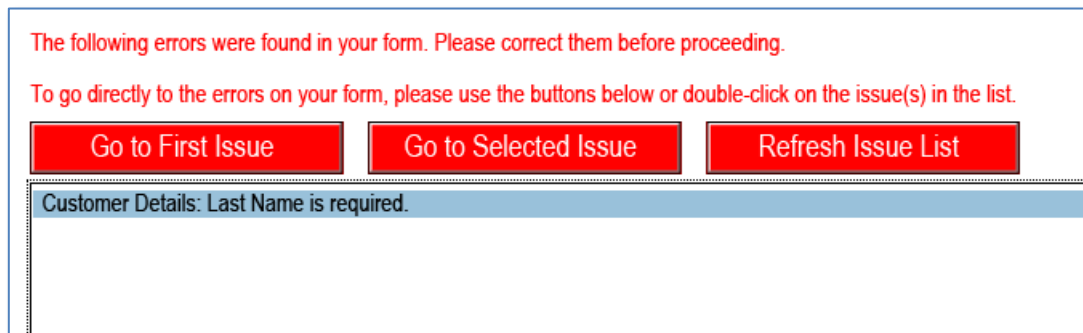


Figure 11 - One way to display errors in a PDF SmartForm. This is the enhanced PDF error handling provided by Composer.

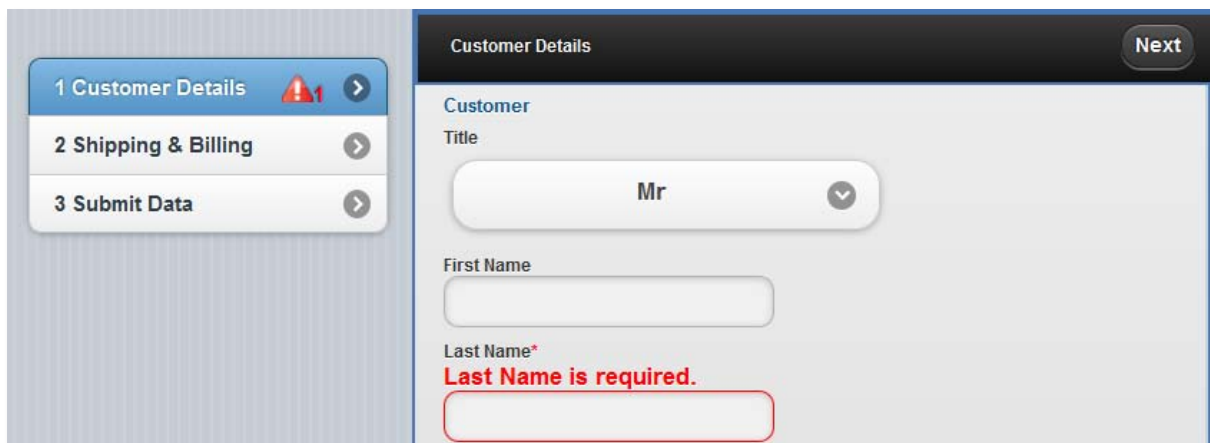


Figure 12 - The same error message, but presented in a way that is more appropriate for a tablet device

The overall forms platform must be capable of two things:

- Allowing you to design a form once, and render it appropriately on each different platform.
- When a user visits your site, the system must be able to “sniff” the user’s device and its capabilities, and serve up the appropriate optimised rendition of the form based on those capabilities.

Although forms are the primary artefact that needs to be designed once and run everywhere, other elements of the system must also run on all types of devices. These include the web-site or portal hosting the forms, a finder application used to find the appropriate forms, and any applications that assist with filling out forms.

SMARTFORM “APPS”

A SmartForm on its own is not enough to enable true replacement of paper transactions. A SmartForm needs to be hosted in an environment that allows the user to participate fully in the overall interaction with the organization.

Once a SmartForm is hosted and configured inside SmartForm Manager, it becomes a SmartForm “App”, which lies somewhere between a simple SmartForm and a full-blown web-application. A SmartForm App supports much more complex interactions than a stand-alone SmartForm, but is much simpler to deploy and manage than a conventional web-application.

These sorts of enhanced interactions that are enabled by Avoka's SmartForm Manager include:

- **Discovery.** Most organizations have many forms, and often it can be difficult and confusing for customers to find the correct form for the transaction they want to complete. The forms platform should provide an intuitive wizard, that allows the user to answer a few simple questions, and be presented with the appropriate forms for the task at hand. The meta-data driving the wizard should be easily configured when the form is deployed.
- **Landing Page.** A landing page may be optionally displayed prior to presenting the form itself. The landing page typically contains information about the purpose of the form, how long it will take to complete, what the user will need in order to complete it, and what steps will occur after submission. This landing page should be configured without any technical skills.
- **Pre-fill.** The ability to pre-fill the form with any information we already know about the customer, as well as relevant other information, which aids the customer to complete the form quickly and accurately. This requires a platform that supports both single-sign-on and integration with backend systems.
- **Profiles.** A user who completes multiple forms should have the ability to set up a user-profile that contains information completed in previous forms. This should be used to automatically pre-populate subsequent forms.
- **Save Online.** For longer forms, a customer needs to be able to save a partially completed form, and come back and finish it later.
- **Attachments.** Often supporting documents are required to complete an application. The system must support the ability to attach these electronically or by traditional mail/fax.
- **Signatures.** The customer should have the ability to sign the form in a number of different ways, including digital or electronic signatures of various kinds, as well as "wet" signatures.
- **Receipts.** A customer should be able to obtain a read-only and printable copy of their submitted information for future reference. The customer should also be able to see a list of all previously submitted forms.
- **Payments.** A customer should be able to easily make a payment associated with the form they have completed.
- **Status updates.** Many form submissions require a payment which customers expect to have integrated into a seamless experience. Once a submission is made, promising customers with online, email and SMS updates regarding the processing of their request can reduce call volumes to a call centre, as well as being a more convenient way for a customer to check on progress.
- **Registration services.** The system must allow users to self-register, reset forgotten passwords, view and edit their profile information, and perform other actions related to their registration. If necessary, single sign-on should be enabled between the main web site and the forms platform.

It is important that these different types of interactions can be very simply configured when the form is deployed, rather than requiring custom and costly changes to application logic. This allows different combinations of interactions to be defined for different types of forms. An assortment of some common interactions is shown on the diagram below.

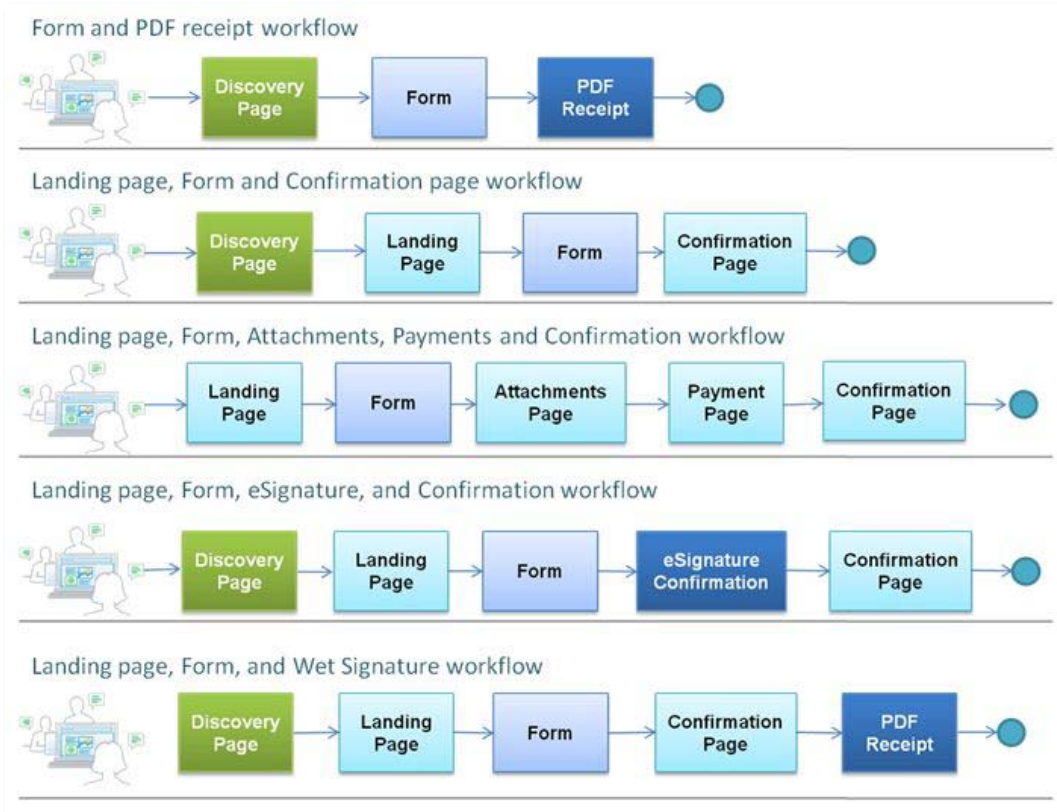


Figure 13 - A set of common interaction patterns

MULTIPLE CHANNELS

All interactions with a SmartForm need to be provided in whatever way makes most sense both for the customer and the organization. This means that multiple channels must be easily accommodated.

Some of the different channels that must be supported are:

- Fixed line vs Mobile devices
- Full screen vs tablet vs small screen devices
- End-users vs Branch employees vs Call Center operators
- Always connected vs occasionally connected users

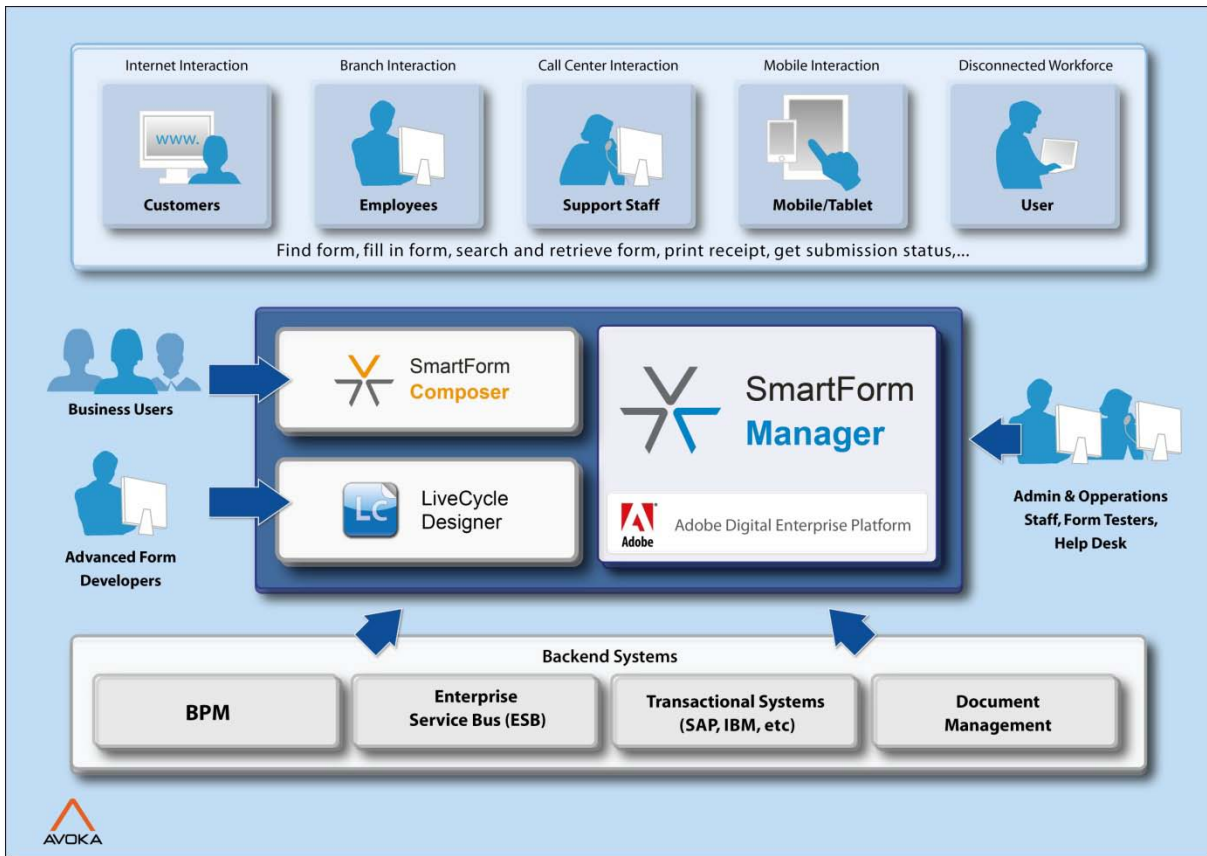


Figure 14 - SmartForm Manager serves as a hub supporting multiple channels of interaction

INTEGRATION WITH EXISTING WEB SITES

There are a number of different patterns in which a SmartForm App should integrate with an existing web site. These patterns include:

- **SmartForm Finder.** The ability to provide a “Finder” application that gives users a customized and guided experience to finding the forms they need.

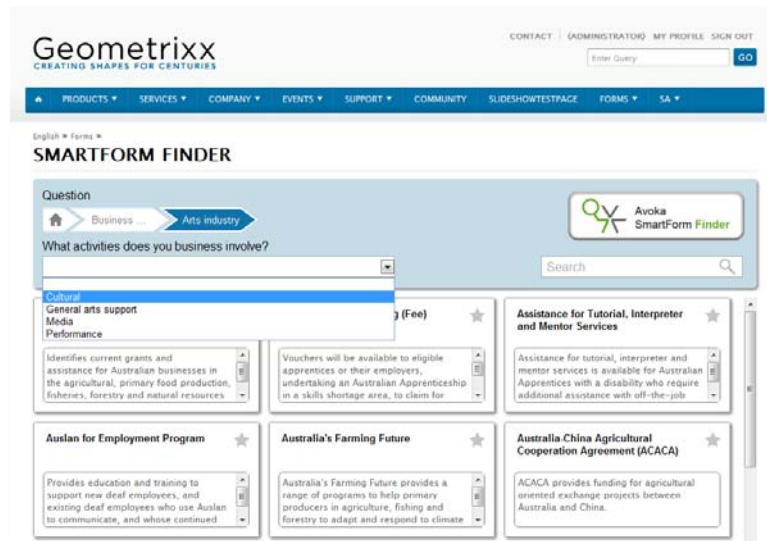


Figure 15 - SmartForm Finder guided user experience

- **SmartForm Web Plugin.** The ability to embed a single form at an arbitrary location in an existing web site. This form could be either embedded in place, or via a link to a pop-up window. It should also be possible to embed other SmartForm App information, such as a list of previously submitted forms.
- **SmartForm Portal.** The ability to provide a customized portal along-side the main corporate site that is optimized for finding, filling and submitting forms.

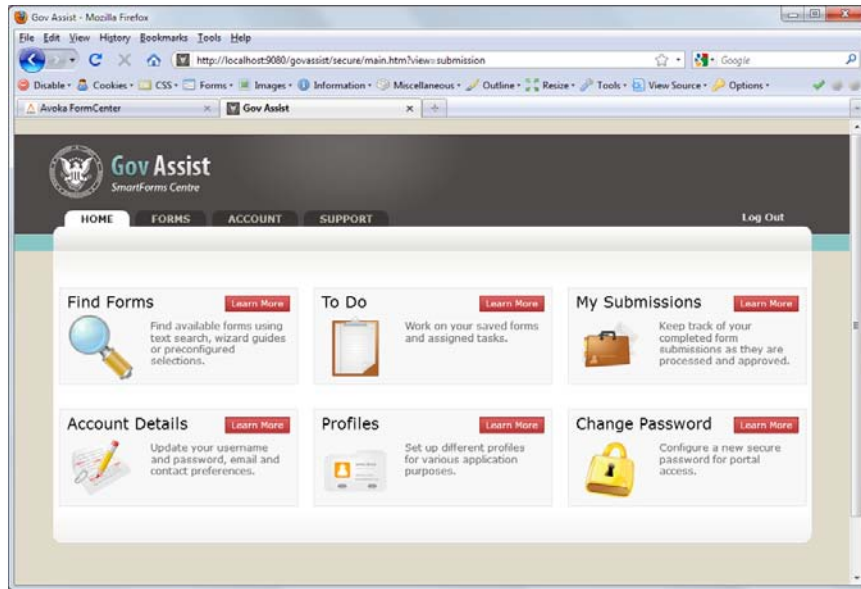


Figure 16 – Customized portal for finding, filling and submitting forms

- **Single-signon.** The ability for a logged-in user to have a seamless experience between navigating the main web site, and the forms.
- **Adobe ADEP Web Experience Management.** Direct and simple integration with sites built using Adobe Digital Experience Platform – Web Experience Management solution (formerly Day CQ Software).

DIFFERENT USAGE MODELS

Different types of users have different requirements for the types of interactions that they have with your forms.

These types of interactions generally fall into three categories.

- **Casual users.** These users generally only fill in a form occasionally. They will usually interact with forms via a custom web portal, or via a form integrated into your existing web site. They may use either a desktop browser or a mobile browser.
- **Registered Users.** Some forms (or some organizations) may require a user to register and authenticate with the site in order to complete a form. This is usually required in order to satisfy security constraints, but also enables additional features, such as pre-fill and save-online. In many cases, there will be single-sign-on between the main web site and the forms platform.
- **Repeat users.** Users who fill in the same or different forms multiple times will enable a better experience by registering with the forms platform, and make use of features such as profiles, form pre-population, and save online. In addition, repeat users on a mobile device may want to download the Self-service App, available on most mobile devices. The Self-Service App allows several useful

features, such as allowing forms to be filled in while offline, pre-population based on profiles, and more.

- **Field workers.** Field workers are different to repeat users in two ways. Firstly, they tend to use the forms platform not just to fill out forms, but also to plan and schedule their activities. Secondly, those activities and their associated forms are often assigned, managed or reviewed by a supervisor, rather than allowing the field worker to simply complete single forms as necessary. There are a large number of features required by field workers completing forms – these are discussed in a separate paper available from Avoka. The Avoka Field Worker solution is a separate software product optimized for the use of Field Worker teams, which depends on and fully integrates with SmartForm Factory.

A USER INTERACTION AND SYSTEM INTEGRATION HUB

Forms don't exist in isolation – they almost always integrate with some sort of backend system. The integration with the backend systems are interwoven with user interactions initiated by the end user. The forms platform must therefore act as a hub, managing the interactions between the user, the form, and the backend systems, at various points in the lifecycle of the overall transaction with the customer.

Any forms platform must be capable of a variety of different types of user-interaction and system-integration, from the trivial to the sophisticated. Some of the types of interactions provide by SmartForm Manager include:

The User Interaction	The System Integration
When the user opens the form ...	<ul style="list-style-type: none"> • “Sniff” the capabilities of the device or platform that the user is using, and present to the user a version of the form that is appropriate for that device. • Pre-populate information into the form based on information we know about the end-user, or other contextual information. • Populate any required drop-down lists or other reference data within the form. • Modify the standard look and feel of the form to be customized to the user’s context. For example, colour schemes and logos may change depending on which organization the user is registered with. This capability is known as white-labelling.
When the user interacts with fields on the form...	<ul style="list-style-type: none"> • Provide ability to search and look-up external information from within certain fields on the form (e.g. address validation). • Usage data should be collected and communicated to the web analytics system to identify fields that may be contributing to abandonment.
When the form is submitted...	<ul style="list-style-type: none"> • Determine whether additional attachments are required, and provide a way for the user to upload those attachments. • Identify if electronic or wet signatures are required, and delay the delivery of the form data until these signatures have been provided. • Provide a capability for payment processes when required and present various supported ways for the user to make the payment prior to delivering the data.
When attachments are submitted...	<ul style="list-style-type: none"> • Ensure that the attachments conform to the supported file types, maximum sizes configured, and check for viruses. • Generate a printable receipt that records all the data

<p>When the form data is delivered...</p>	<p>that the user entered.</p> <ul style="list-style-type: none"> • Send an email to a designated user or group of users containing the form data. This is the simplest type of integration, which allows for “swivel-chair integration” i.e. copy/paste into other systems. • Invoke a workflow or business process with the form and data. SmartForm Manager provides built-in integration for Adobe’s Digital Enterprise Platform – Process Management, but other BPM systems are also supported. • The system should store all or some of the data from the form into custom tables within a database, and provide the ability to produce reports from this data. • Archive a read-only copy of the form on the file system or an enterprise content/document management system.
<p>When the user views their submission history...</p>	<ul style="list-style-type: none"> • Display the current status of their overall transaction.
<p>When the overall transaction is complete...</p>	<ul style="list-style-type: none"> • The system should be a configured to enforce how long the user’s data is retained, or whether it is immediately deleted.
<p>When a new version of the form is deployed...</p>	<ul style="list-style-type: none"> • The system should start serving up the new version of the form from its designated start date – but continue to use the old form for historical transactions.

Some of these different types of interactions are shown in the diagram below:

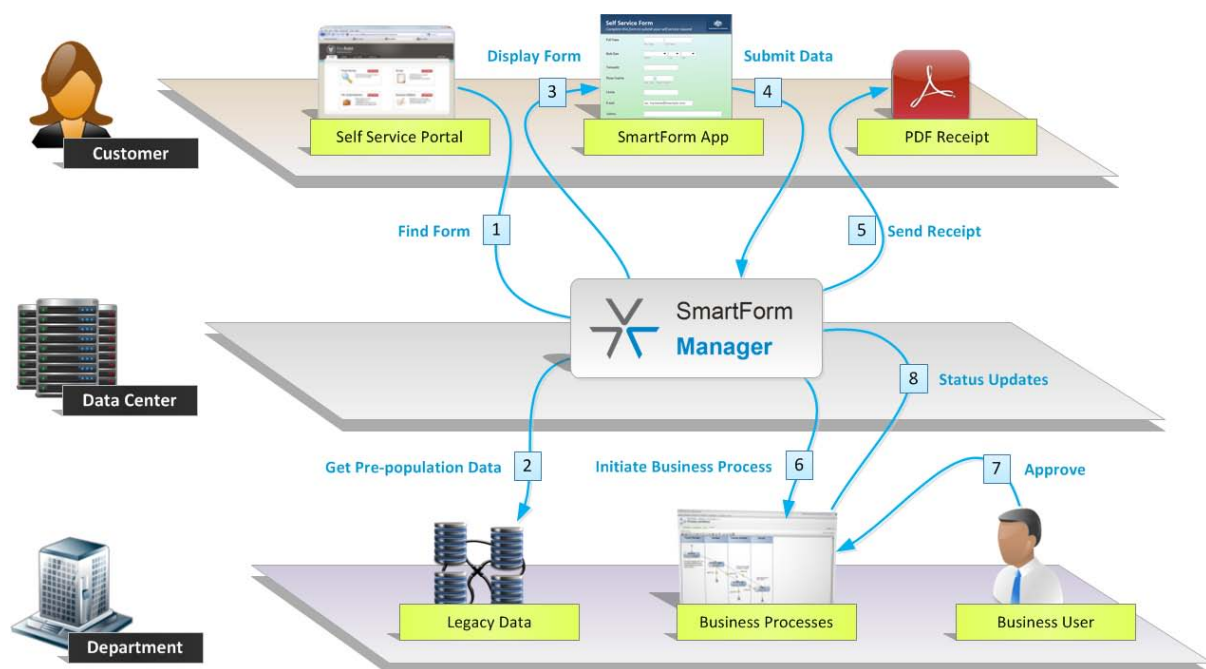


Figure 17 - Displays some of the types of interactions between users and SmartForm Manager, which in turn result in integrations with backend systems

INTEGRATION CHALLENGES

A secondary integration issue often occurs because of the way that the IT department configures the servers and network. The forms platform is often hosted in the “DMZ” or public zone (or sometimes in the cloud), whereas the systems that need to be integrated to are in the private zone or “behind the firewall” . The private zone can see the public zone, but the public zone is prevented from communicating directly with the private zone to ensure security. This makes it difficult for forms that are submitted by users to inject their data directly into backend systems.

In order to overcome this issue, SmartForm Factory includes an Integration Agent. This Agent runs in the private zone, and periodically “pulls” submissions from SmartForm Manager running in the public zone, thus safely bypassing the security issues. The integration agent then communicates with the backend systems directly.

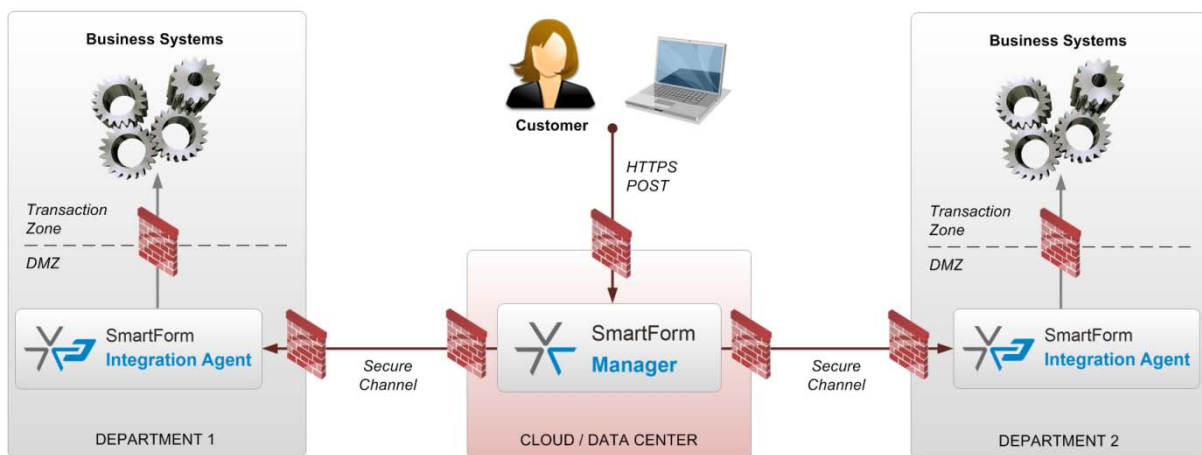


Figure 18 - Integration Agent Architecture. This diagram shows a single instance of SmartForm Manager in the cloud interacting with two different departments, each within their own private zones.

OTHER CAPABILITIES

A robust and mature SmartForm platform must provide a number of additional functional and non-functional capabilities. These include:

- Scalability.** The SmartForms platform must ensure that it scales well to ensure that users are serviced, particularly in times of peak load. Peaks can often be much higher than normal levels, due to submission deadlines, whether this be registering to vote or prior to an election or submitting an application for a product or service to take advantage of a special offer. The SmartForm platform must be able to scale both horizontally and vertically, to take advantage of available hardware and infrastructure. The system should also be able to easily take advantage of additional servers being brought on-line in a cloud or cloud-like environment.
- Quality of Service.** The SmartForms platform should provide guaranteed levels of service. At some level of load, no matter how much hardware is in production, the load may exceed capacity. Should this level be exceeded, rather than providing uniformly poor response to everyone, the system should instead start diverting overload users to a static “Come back again later” page. This ensures that those users that are serviced will achieve adequate responsiveness and will be able to complete their tasks. Lower priority tasks such as delivering data should be deferred until the load decreases. The system should be able to monitor itself rather than requiring human intervention to detect overload situations.

SmartForm Factory Quality of Service

- Excessive Load Protection



Figure 19 - SmartForm Manager Quality of Service and Prioritization

- **Metrics.** The SmartForms platform should gather and report on a large number of metrics that will allow the IT department to analyse and plan. For example, metrics about browsers, platforms, and versions will allow the IT department (and the business) to decide what client platforms are most important to cater for when designing forms. Metrics relating to each form, including requests and submissions, allow administrators to determine which forms are most highly used, abandonment rates, and peak periods. Other metrics are provided that are useful to line of business users, Marketing and IT.
- **Analytics.** Just like any other well-behaved web application, the forms platform should gather analytical information about the usage of forms, at the 'form', 'page' and even the individual 'field' level. Of particular importance are the reasons for abandonment, because an abandoned form generally means a lost opportunity. At a minimum, abandonment means that the form will be submitted via another more costly channel. The forms platform should collect basic analytics information without any additional software, but should also integrate with standard analytics engines such as the Adobe Online Marketing Suite powered by Omniture, allowing form metrics to be integrated into overall web site metrics.
- **Security and Privacy.** The forms platform must be inherently secure, and protect data. Much of the overall security of a system is based around the firewalls, servers and intrusion and other security devices that are installed and configured around the platform – but the core platform should be architected and built to enable good security and privacy.
 - **Flexibility.** The forms platform should be a mature offering that has grown over several years based on the needs of many different types of organizations. This will ensure that it is flexible enough to be configured to suit your unique needs.
 - **Manageability.** It is vitally important that the forms platform cater to the needs of system administrators as well as Form Developers and business people. The system administration console should provide a variety of capabilities, including:
 - **Ability to easily** deploy a new form, or a new version of an existing form
 - Easily configure the user interactions associated with a form
 - Metrics around form usage, and user demographics
 - Errors and logs
 - System maintenance

- User maintenance
- Security configuration
- And more...