Global Forms Service

Functional Specification

FINAL

Revision History

Date	Level	Revision
Aug. 20	1.1	• Diagram showing “integration hub” concept. • Interaction diagram to show service wrapping.

July 2009

Table of Contents

Overview

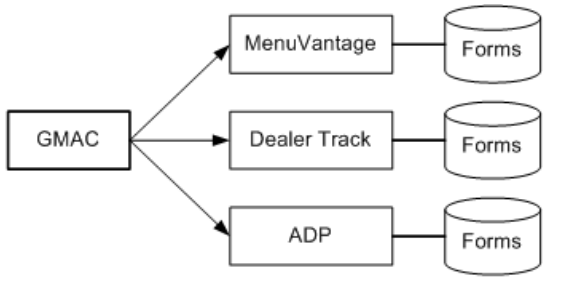
In automotive retail, many providers of Finance and Insurance (F&I) products have the ability to produce contract forms in real time, at the point of sale. GE Capital, for instance, can produce a vehicle service contract that is validated, pre-filled, and ready for signature.

For best results, this functionality will not be confined to the provider's web site. It will be exposed as a web service, and made available for integration with all relevant point-of-sale (POS) systems. The purpose of a "global forms service" is to make this approach available to all forms providers on an outsourced basis.

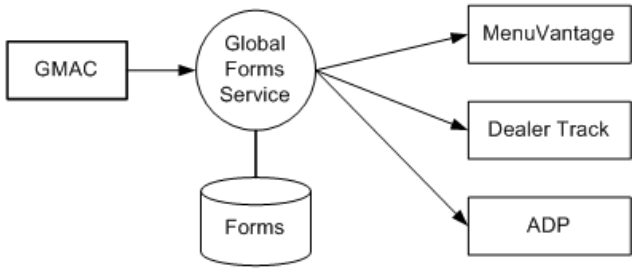
In any industry, providers of such functionality face a common challenge when it comes to forms management. Exposing validation services, integrating with outside systems, and originating contracts are all "core" functions unique to the given provider. Forms management, however, is a cumbersome and non-core function ideally suited to outsourcing.

Forms providers are typically unwilling to relinquish control of their forms, and they certainly do not want them to be used without control over the validation logic. As a result, the forms functionality is confined to the provider's web site and a limited number of POS developers with whom the provider is willing to share forms.

In the example below, an automotive F&I provider supplies forms to three POS systems. Each system must receive updates for new forms and rule changes. By outsourcing the integration work, the provider avoids duplicate efforts and ensures consistent results.



Forms provider manages multiple POS relationships.

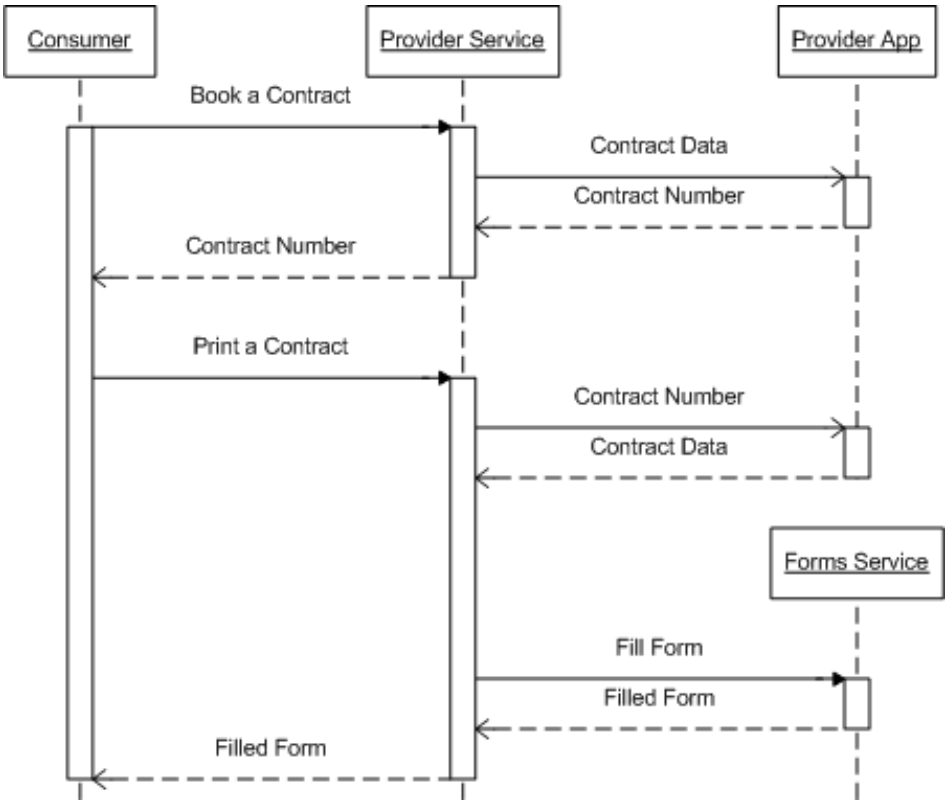


Forms service handles form requests automatically.

There are millions of web pages that expose form-filling directly to the customer, using exactly two techniques. The user is presented with either a fillable Adobe™ form, or a purpose-built web page. Both of these techniques have disadvantages in terms of scalability, usability, and data access – and neither of them is readily available for systems integration.

A service-oriented approach is needed, which would expose forms for integration via web service. In addition, this would support direct access by customers, encapsulating the data collection, and generating up-to-date web forms automatically.

The basic requirement is for a web service to return the pre-filled contract as a PDF or other portable format. The forms provider exposes a second service to third-party applications, encapsulating details of the contracting system and the forms library. Consider the interaction depicted below.



Here, the forms service exposes a web method to send the pre-filled PDF back to the consuming application. The provider wraps this method with his own, exposing a common web service to the consumer.

This is the ideal sequence, in which the forms service is encapsulated by the provider's own service. The *Functional Specification* below is based on this sequence. As needed, the forms service may also call the contracting system directly, or be called directly by the consuming application.

Scope

This document describes the design of a "global" forms service (GFS). The name is meant to suggest that a single such service may be used for all form providers, as described above, regardless of application. The scope of the system includes:

1. The web service
2. The forms library
3. An administrator's interface
4. A form-description tool
5. A generic web page

The generic web page consumes the form-filling service and dynamically creates input controls for the required data elements. This page will be used for testing the web service, and it may also be accessed by consuming applications. In this same vein, the administrator's interface will communicate with the forms library via web service, so that the provider may use administrator functions in some other context.

The provider is intended to expose the forms service to third-party applications. Billing can be based on forms rendered, forms hosted, forms uploaded – or some combination thereof. The billing requirements are described later in this document.

Definitions

- **Forms Provider** – This business entity is a customer of the Global Forms Service. They have a need to provide form-filling services to their customers, which they outsource to the GFS.
- **Consumer** – This business entity is a customer of the Forms Provider. They support the Consuming Application.
- **Consuming Application** – This application "consumes" the forms service, in the web-service sense. The GFS is "exposed" to the consuming application, to provide form-filling on behalf of the provider.
- **Forms Administrator** – The provider's forms administrator (not the GFS system administrator). This is the user of the administrator's web site.
- **User** – This is the operator entering data, either into the consuming application, or directly to the GFS via its dynamic web page.

Market Analysis

As one would expect, there are millions of forms in circulation. A Google search on "forms library" finds 38 million pages, mostly in health care and government. Each page represents several forms – scores of forms, in the case of government agencies.

These web sites employ various approaches to the form-filling problem. The approaches fall into two categories. Where data capture is desired, the site presents a customized web form. Otherwise, as web forms are costly, the site presents an Adobe™ PDF. The PDF may be fillable, but there is no data capture. Users typically are requested to fill the form and submit it via fax.

- Tri-Care Health:
  - Non-fillable PDF
  - Fillable documents in MS-Word 2003 format
  - Customized web form
- The GSA has a well-organized forms library, with attention to "frequently used" forms:
  - Fillable PDF – does not capture data
  - Accessible Form Net – custom web form with ADA features
  - Form Flow – for pen-based systems
  - Form Docs – requires proprietary file format and client installation
- MetLife – fillable PDF
- University of Michigan – web forms and non-fillable PDF
- AmeriCorps – non-fillable PDF and Word 2007
- Department of Labor – hundreds of non-fillable PDF

Competing Solutions

There are some modern, service-oriented systems in the forms management space. The likely reason these are not in evidence on the big, public forms sites is that they require forms to be authored using a proprietary technology. Most of the systems in this space are primarily form development systems, a market dominated by Adobe. A competitive advantage for the GFS process is that it accepts legacy PDF files as input.

- **infoRouter** – This sounds like a great system, but it is designed for in-house workflow management. The focus is not on integration with outside systems:

  > **…** capable of rendering documents data in multiple formats. Users can enter document data using a friendly HTML form template. infoRouter stores the form values in an XML file.

- **DocFinity** – Another good system, again aimed at in-house form developers. Produces Adobe FDF files from an XML-based forms authoring tool.
- **Form Flow** – for pen-based systems.
- **Form Docs** – requires proprietary file format and client installation.

Adobe Form Designer

Adobe Form Designer allows the user to generate an HTML web form and a fillable PDF from the same template, but not a web service. This allows a choice of data capture techniques at the user interface, but only for new forms. The GFS approach includes technology for adapting legacy forms – including non-tagged PDF forms – and generating web services. Results of the competing approaches may be summarized graphically as follows:

Adobe Form Designer

- Design Template → Fillable PDF
- Design Template → HTML Web Form

Global Forms Service

- Legacy PDF → Fillable PDF
- Legacy PDF → HTML Web Form
- Legacy PDF → Web Service

The GFS has the further advantage of not being tied to Adobe. If there is ever a need to support forms rendered as GIF, or some future format, the overlay approach is easily adapted. Over time, we can expect new file formats to evolve.

The Global Forms Service will be sufficiently general to support its basic functions of data capture, web-form generation, and form filling, using any file format. If a provider wishes to change their file format, we can negotiate a fee for re-mapping the forms.

Revenue Model

This is a low-cost, low-margin business with straightforward technology. The challenge lies in achieving sufficient volume to cover the payroll expense. A sample analysis is given in the appendix. This assumes a monthly fee of $15 per form. Under these assumptions, the model becomes profitable around 10,000 forms. Total investment is roughly $300,000.

A likely GFS customer would be a forms provider who:

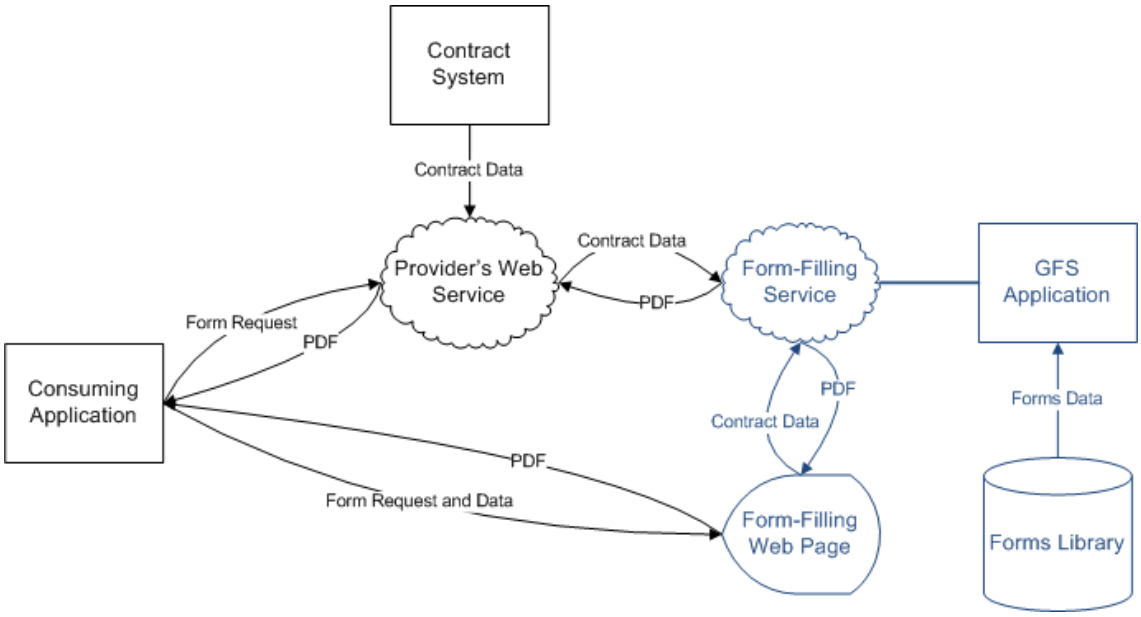
1. Already has a web service for data capture, but does not wish to convert and maintain their forms (the GMAC example).
2. Issues forms to POS partners, and wishes to avoid managing multiple relationships.
3. Uses Adobe forms and wishes to add a data capture capability.
4. Uses web forms for data capture and finds this to be expensive.

Examples of all four classes can be found in automotive F&I. Unfortunately, the total population in this market is unlikely to reach the needed volume. Research on the web shows the third class to be by far the most numerous, easily achieving the 10,000 forms mark. For most of the market – classes 1 through 3 – a key value-add is the ability to convert legacy forms into the GFS model.

We have specialist knowledge regarding integration requirements for automotive F&I. Other industries may be presumed to have similar requirements. For instance, POS integration for health care is likely to be popular.

Functional Specification

The diagram below shows the "front end" of the Forms System. That is, the part used to produce forms on behalf of the provider. For clarity, the forms-system components are shown in blue. Here, the provider supplies contract data to the forms service, to be used in filling the form. Also shown is an alternative, wherein the consuming application provides data directly to the Forms System.



GFS Application

This is the core web application. It consists of a web site and the business objects needed to support the form-filling service. The usual multitier design practices apply: the web site and services only communicate with the business layer, the business layer only communicates with the database by way of stored procedures, etc.

Form-Filling Service

Each form will be stored with a "data map" containing form-filling instructions. The data map identifies each data element to be placed on the form, and where to position it. There are two popular techniques for this latter requirement: position given by x-y coordinates (the "overlay" method) and position given encoded within the form using a proprietary Adobe™ standard.

The form-filling service will support either technique, for a given form, according to the provider's preference. In either case, the data map will be stored as XML in the forms library. Examples of both techniques are given in the Appendix. Methods supported by the service are:

- Get Blank Form – Return blank PDF from the forms library.
- Get Blank Data Map – Return data map from the forms library.
- Get Filled Form – Accept data from data map, and return a filled form.
- Various query methods, such as "form valid for this date?" and "size of paper?"

The data map will also include coordinates for a signature box, in case the consuming application has the ability to collect an electronic signature. Multiple boxes for the signer's initials may also be included.

Form-Filling Web Page

The form-filling page consumes the web service, above. It serves as a test page for the service, and it may also be launched (using SSO) from a consuming application. Text boxes on the form-filling page are built dynamically using the data map of the desired form. For this purpose, the data map will include some rudimentary formatting information. This page does not access the provider's contracting system and so it cannot issue a contract number. It can, however, issue a tracking number according to provider-specific rules.

The consuming application will pass a form identifier with the launch, and may also pass any data available to be pre-filled on the page. The page should employ variable style sheets, to emulate the look and feel of the consuming application. When the page exits, it will return the filled form and data map to the consuming application.

Data Map Development

Developing the data map for each form is a time-consuming effort, and represents much of the value added by the GFS. The mapping technician will be aided by a utility program which displays plot points on the forms, prompts for element names, and generates the data map.

For each data element, the map will contain a short name, a label, and a data type sufficient to generate the form-filling page, above, dynamically. Developers of consuming applications need only develop a single "mapper class" to align their data models with those of their providers. Sample data map syntax is given in the Appendix.

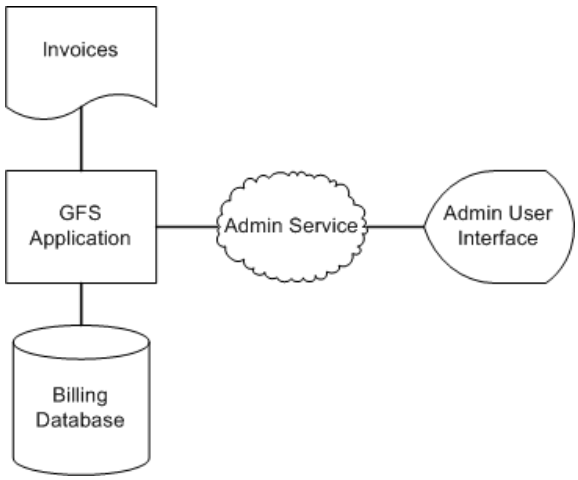
The GFS will not perform any data transformation, because it does not have direct access to the provider's data model. For example, if the consuming application has a SQL Date element and the form is expecting a two-digit year, the application will have to send a two-digit year.

Forms Library

The forms library database is the central repository of providers' blank forms and data maps. See *System Design*. It also supports administrative functions for the provider, including the processing status of new forms. Note that the database uses an internal form ID distinct from the provider's "form code." Each provider may use its own convention for form identification. Consuming applications will have obtained a form code from the provider's application (a contracting system, for instance) before invoking the Global Forms Service.

Billing Database

The diagram below shows the "back end" of the Forms System. The "administrator" referenced is the provider's forms administrator, not the GFS system administrator. This user interface is shown decoupled by a web service, in case the provider wishes to use a different user interface.



Every rendering event is logged in the GFS database, and invoicing is supported by a "billing view." This supports various payment plans. The simplest is for the provider to pay a flat monthly fee for hosting each form, plus an initial charge for loading it. Other plans may require the provider, or the consumer, to pay for each rendering event.

Administrative User Interface

The provider is able to perform various administrative functions:

- Upload a new form, and assign an effective date for it.
- Decommission an old form, by assigning a retirement date.
- Modify the assignment of form codes.
- Administer consumer logins and permissions.
- Query usage of forms (and preview the bill).
- Query the workflow status of a new form.

Newly-uploaded forms automatically enter the data-mapping work queue. Providers should allow five business days for new forms to be mapped.

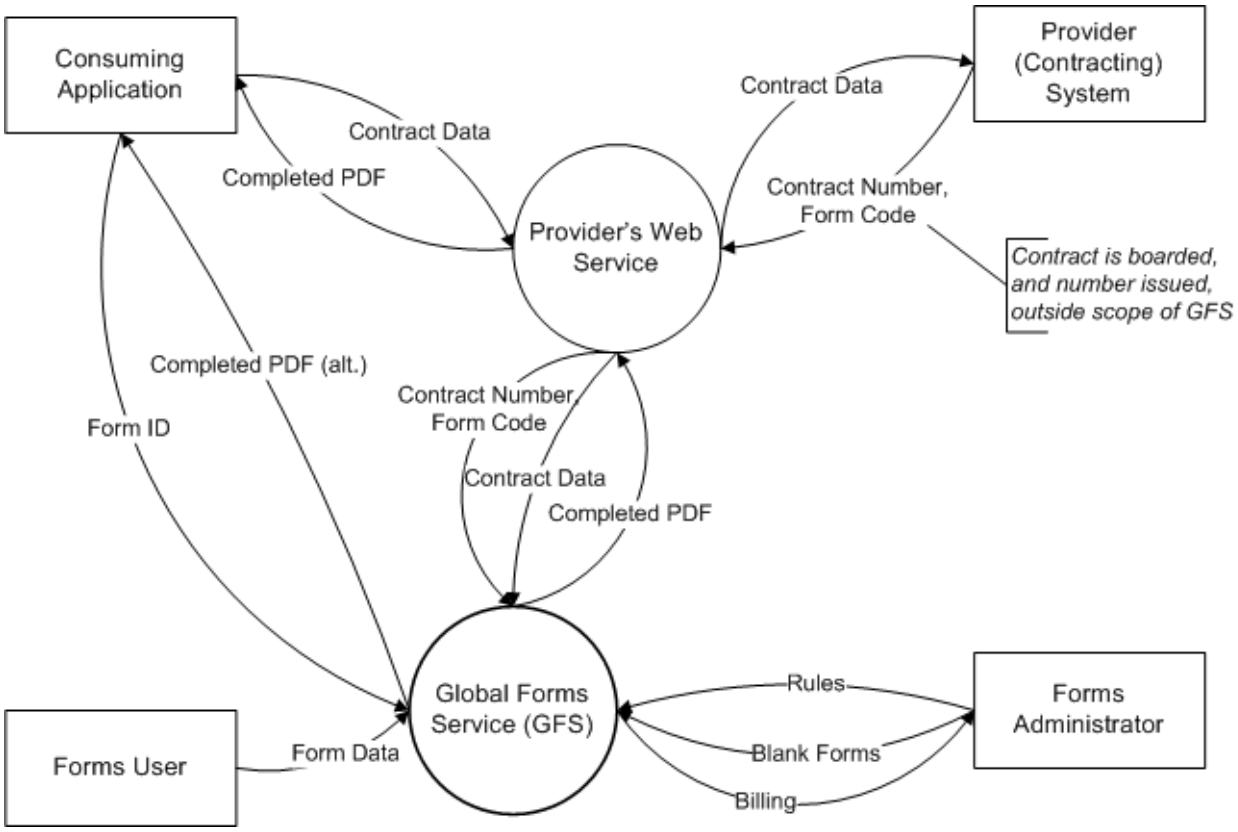
Administrative Service

This web service decouples the administrative user interface from the GFS application. That is, all of the functions listed above must proceed through a web service. By consuming the web service, the forms administrator may develop his own administration system or incorporate GFS functionality into a legacy system.

System Design

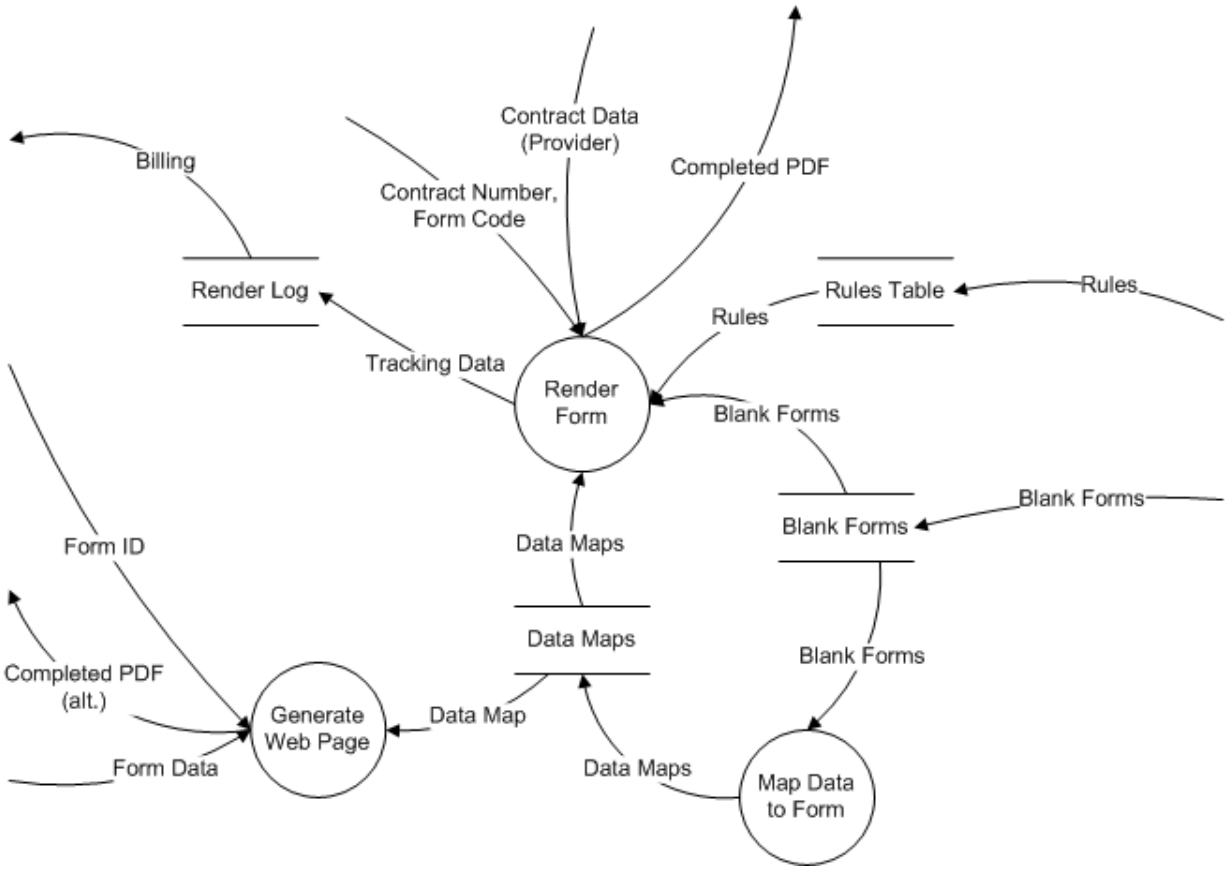
Context Diagram

This diagram shows how the GFS serves forms in conjunction with the provider's systems and forms administrator. Also shown is the alternative flow wherein the consuming application passes control directly to the GFS dynamic web page.



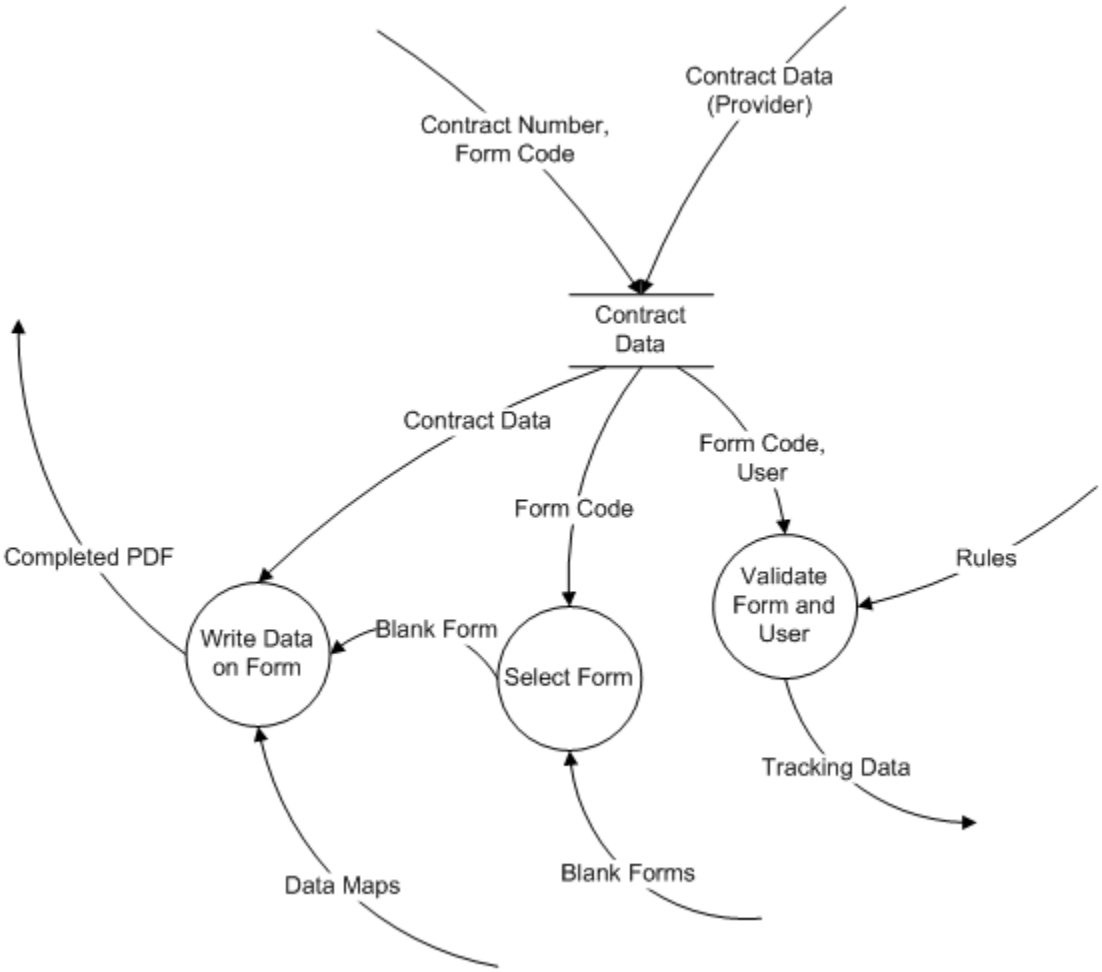
Level One Diagram

This diagram expands the GFS bubble to show how data flows through, and within, the system. Blank forms are submitted into a work queue by the forms administrator, who has access to a processing-status indicator.



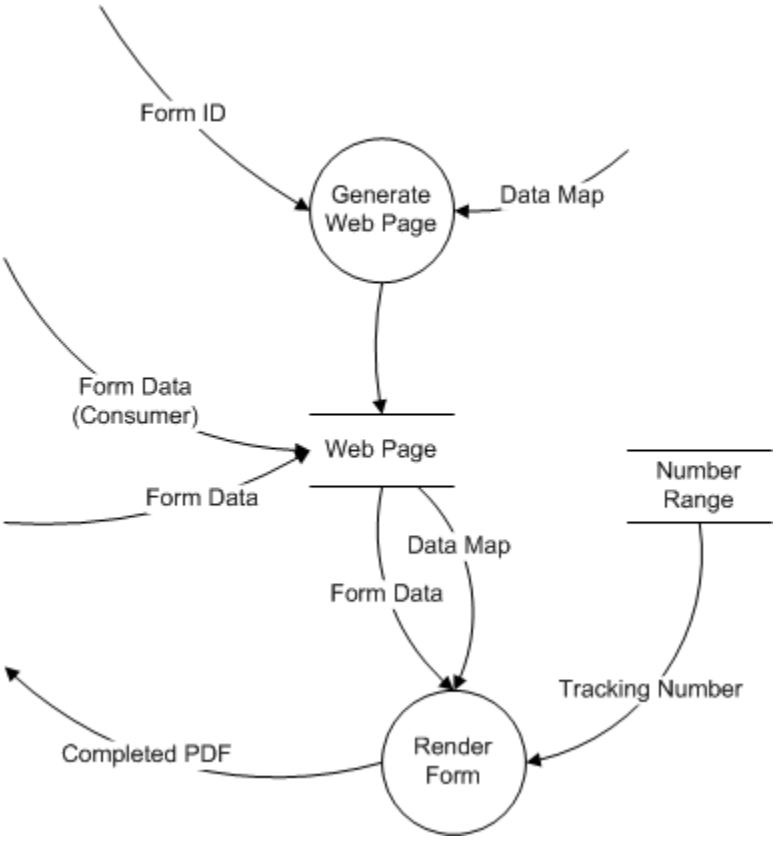
Render Form

This diagram expands the "render form" bubble to show the discrete flows to validation, tracking, form lookup and rendering. "Render form" is shown at this level and is also called within "generate web page."



Dynamic Web Page

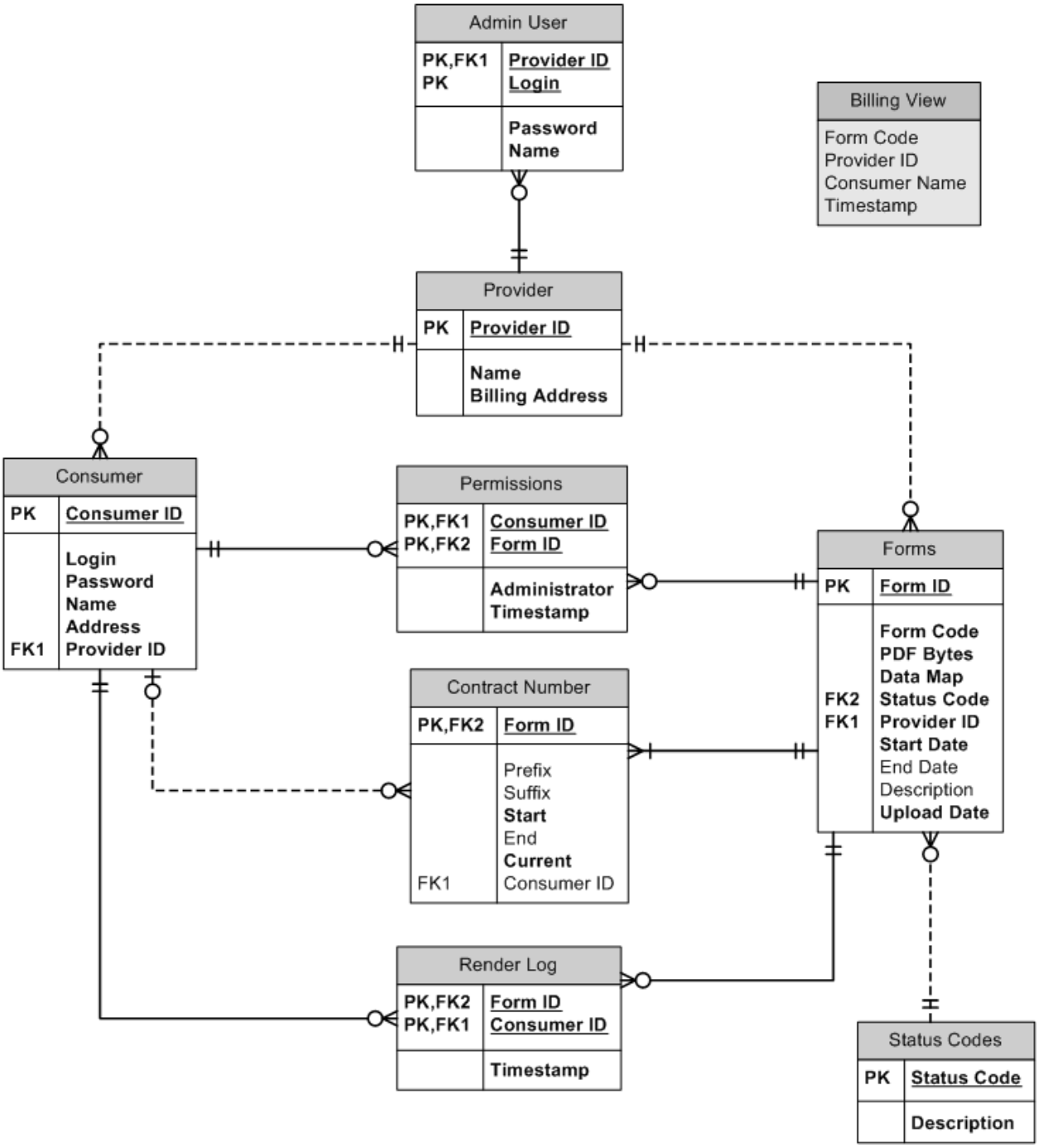
This is the alternative flow wherein the user interacts directly with the GFS by means of its dynamically-generated form filling page. In this case, there is no contract number but the page assigns a "tracking" number from a specified range.



Data Dictionary

Flow Name	Definition	Sample Elements
Contract Data	Application specific data for the given form.	Name, address, contract price.
Completed PDF	Printable form in PDF (or other image) format, with data filled in.	PDF bytes.
Contract Number	Application specific number, issued by provider's system.	Integer or string, probably having some intelligent syntax known only to the provider.
Tracking Number	Application specific number, generated by the GFS according to provider syntax.	Sequential integer plus a prefix or suffix that can vary with the consumer.
Form Code	Provider's identifier for the form.	Integer or string, probably having some intelligent syntax known only to the provider.
Form Data	Same as contract data, but input directly to the form without validation by the provider's system.	Name, address, contract price.
Rules	Access rules for each form, identifying allowable consumer systems.	Login ID, password, form ID, effective date.
Blank Forms	Printable form in PDF (or other image) format, without data.	PDF bytes.
Billing	Billing information transmitted to provider.	Form counts, rendering counts, balance due.
Tracking Data	Log info for each rendering event.	Consumer ID, form ID, timestamp.
Data Map	XML document indicating where to place each data element on the form, and also how to format it on the form-filling page.	Tags for form fields or x-y coordinates, and data element names.

Data Model


Admin User
PK,FK1 Provider ID
PK Login
Password
Name
Billing View
Form Code
Provider ID
Consumer Name
Timestamp
Provider
PK Provider ID
Name
Billing Address
Consumer
PK Consumer ID
Login
Password
Name
Address
FK1 Provider ID
Permissions
PK,FK1 Consumer ID
PK,FK2 Form ID
Administrator
Timestamp
Forms
PK Form ID
Form Code
PDF Bytes
Data Map
FK2 Status Code
FK1 Provider ID
Start Date
End Date
Description
Upload Date
Contract Number
PK,FK2 Form ID
Prefix
Suffix
Start
End
Current
FK1 Consumer ID
Render Log
PK,FK2 Form ID
PK,FK1 Consumer ID
Timestamp
Status Codes
PK Status Code
Description

Appendix

Revenue Model

	Compensation			Overhead		Expenses	Customers		Forms		Fees			Income
Month	Devel.	Sales	Support	Hosting	Office		New	Total	New	Total	Setup	Maint. Fees	Comm.	Margin	Monthly
1	$15,000	$0	$0	$3,000	$3,000	$24,333	0	0	0	0	$0	$0	$0	$0	($24,333)
2	$15,000	$0	$0	$3,000	$3,000	$24,333	0	0	0	0	$0	$0	$0	$0	($24,333)
3	$15,000	$0	$0	$3,000	$3,000	$24,333	0	0	0	0	$0	$0	$0	$0	($24,333)
4	$15,000	$0	$0	$3,000	$3,000	$24,333	0	0	0	0	$0	$0	$0	$0	($24,333)
5	$15,000	$0	$0	$3,000	$3,000	$24,333	0	0	0	0	$0	$0	$0	$0	($24,333)
6	$15,000	$0	$0	$3,000	$3,000	$24,333	0	0	0	0	$0	$0	$0	$0	($24,333)
7	$15,000	$10,000	$0	$3,000	$3,000	$34,333	1	1	100	100	$10,000	$1,500	($4,000)	$2,500	($31,833)
8	$15,000	$10,000	$0	$3,000	$3,000	$34,333	2	3	200	300	$20,000	$4,500	($8,000)	$6,500	($27,833)
9	$15,000	$10,000	$0	$3,000	$3,000	$34,333	3	6	300	600	$30,000	$9,000	($12,000)	$12,000	($22,333)
10	$15,000	$10,000	$5,000	$3,000	$3,000	$39,333	4	10	400	1,000	$40,000	$15,000	($16,000)	$19,000	($20,333)
11	$15,000	$10,000	$5,000	$3,000	$3,000	$39,333	4	14	400	1,400	$40,000	$21,000	($16,000)	$25,000	($14,333)
12	$15,000	$10,000	$5,000	$3,000	$3,000	$39,333	4	18	400	1,800	$40,000	$27,000	($16,000)	$31,000	($8,333)
13	$15,000	$10,000	$10,000	$3,000	$3,000	$48,167	4	22	400	2,200	$40,000	$33,000	($16,000)	$37,000	($11,167)
14	$15,000	$10,000	$10,000	$3,000	$3,000	$48,167	4	26	400	2,600	$40,000	$39,000	($16,000)	$43,000	($5,167)
15	$15,000	$10,000	$15,000	$3,000	$3,000	$53,667	4	30	400	3,000	$40,000	$45,000	($16,000)	$49,000	($4,667)
16	$15,000	$10,000	$15,000	$3,000	$3,000	$53,667	4	34	400	3,400	$40,000	$51,000	($16,000)	$55,000	$1,333
17	$15,000	$10,000	$15,000	$3,000	$3,000	$53,667	4	38	400	3,800	$40,000	$57,000	($16,000)	$61,000	$7,333
18	$15,000	$10,000	$20,000	$3,000	$3,000	$59,167	4	42	400	4,200	$40,000	$63,000	($16,000)	$67,000	$7,833
19	$15,000	$10,000	$20,000	$3,000	$3,000	$59,167	4	46	400	4,600	$40,000	$69,000	($16,000)	$73,000	$13,833
20	$15,000	$10,000	$25,000	$3,000	$3,000	$64,667	4	50	400	5,000	$40,000	$75,000	($16,000)	$79,000	$14,333
21	$15,000	$10,000	$25,000	$3,000	$3,000	$64,667	4	54	400	5,400	$40,000	$81,000	($16,000)	$85,000	$20,333
22	$15,000	$10,000	$25,000	$3,000	$3,000	$64,667	4	58	400	5,800	$40,000	$87,000	($16,000)	$91,000	$26,333
23	$15,000	$10,000	$30,000	$3,000	$3,000	$70,167	4	62	400	6,200	$40,000	$93,000	($16,000)	$97,000	$26,833
24	$15,000	$10,000	$30,000	$3,000	$3,000	$70,167	4	66	400	6,600	$40,000	$99,000	($16,000)	$103,000	$32,833

PDF Overlay Tags

```
<PlotPoint>
<X>410</X>
<XBound>700</XBound>
<Y>707</Y>
<YBound>100</YBound>
<PageNumber>-1</PageNumber>
<LayerNumber>1</LayerNumber>
<FontName>Arial</FontName>
<FontSize>10</FontSize>
<FieldType>0</FieldType>
<PreviewDisplay>ContractNo</PreviewDisplay>
<DataSource>ContractNumber</DataSource>
<XPath />
<HTMLTags />
<SingleCharacter>-1</SingleCharacter>
<Formatting />
<CkBoxDisplay />
<CkBoxCompareValue />
<CkBoxCustomGroupingCode />
<CkBoxSequence>1</CkBoxSequence>
</PlotPoint>

<PlotPoint>
<X>80</X>
<XBound>700</XBound>
<Y>621</Y>
<YBound>100</YBound>
<PageNumber>-1</PageNumber>
<LayerNumber>1</LayerNumber>
<FontName>Arial</FontName>
<FontSize>10</FontSize>
<FieldType>0</FieldType>
<PreviewDisplay>888888888</PreviewDisplay>
<DataSource>ZipCode</DataSource>
<XPath />
<HTMLTags />
<SingleCharacter>-1</SingleCharacter>
<Formatting>StringHelper.FormatZipCode(<Value>, false)</Formatting>
<CkBoxDisplay />
<CkBoxCompareValue />
<CkBoxCustomGroupingCode />
<CkBoxSequence>1</CkBoxSequence>
</PlotPoint>
```

Adobe FDF

```
<fields xmlns:xfdf="http://ns.adobe.com/xfdf-transition/">
  <field xfdf:original="c1_01(0)">Off</field>
  <field xfdf:original="c1_02(0)">Yes</field>
  <field xfdf:original="c1_03(0)">Off</field>
  <field xfdf:original="c1_04(0)">Off</field>
  <field xfdf:original="c1_05(0)">Off</field>
  <field xfdf:original="c1_06(0)">Off</field>
  <field xfdf:original="f1_01(0)">Mark Virag</field>
  <field xfdf:original="f1_02(0)">Virag Consulting, Inc.</field>
  <field xfdf:original="f1_04(0)">1994 E Sunrise Blvd., Suite 248</field>
  <field xfdf:original="f1_05(0)">Fort Lauderdale, FL 33304</field>
  <field xfdf:original="f1_06(0)" />
  <field xfdf:original="f1_17(0)">92</field>
  <field xfdf:original="f1_19(0)">0183337</field>
</fields>
```