AppsFidelity 3.2

User Manual

June 29, 2015

Genii Software Ltd.
http://www.GeniiSoft.com

Table of Contents

1

Overview

This chapter will discuss what AppsFidelity *is, what business problems it is designed to solve, and how it goes about solving them.*

Description

AppsFidelity is a specialized software extension that runs on the IBM Domino server, and optionally on the IBM Notes client, using IBM's "extension manager" technology to seamless editing integration between the IBM Notes client and various third party rich text editors which run in a web browser. No scripts or agents are required.

Purpose

AppsFidelity was developed to allow users to edit rich text fields in both the Notes client and in a web browser without significant loss of fidelity when switching between them for editing or viewing.

AppsFidelity works by dynamically converting between Notes rich text and HTML/XHTML on demand, with specific attention to storage of shared elements so that loss in formatting and content is minimized. Graphics, tables, attachments and links may be edited on either the Notes client or on the web, depending on which third party web editor is used. AppsFidelity simply makes sure the conversion is as seamless as possible and minimizes the loss usually encountered in such a conversion.

Compatibility

AppsFidelity 3.2 on the server supports all point releases of IBM Domino 8.5.x/9.x on Windows 32-bit/64-bit and Linux. XPage support requires IBM Domino 8.5.3 or higher. Other platforms will be supported if demand warrants their addition.

AppsFidelity 3.2 on the client is available for IBM Notes 8.5.x/9.x on Windows and Linux directly and on the Mac indirectly through the server.

XPage support requires IBM Notes 8.5.3 or higher.

AppsFidelity 3.2 in the web browser can support a variety of third party rich text editors, and can be easily adapted to support others based on a set of configurable characteristics, but is primarily configured and designed for use with the

integrated CKEditor. Browser support is mostly dependent on the support necessary for the rich text editor.

Just in time conversion

To limit the inevitable small losses created by converting between two formats, AppsFidelity only converts from rich text to HTML or MIME and back when the conversion is needed. If a document is mostly edited and accessed through the Notes client, the conversion to HTML or MIME does not need to happen each time, but will be done just before the web browser needs to display the content. Similarly, if a document is mostly edited or accessed through the web or XPages interface, the conversion to rich text does not need to happen each time.

Detection & Optimization

AppsFidelity will detect whether a field should be converted based on trigger conditions. This allows AppsFidelity to identify changes made and replicated into the database as well as those made on the client. It also allows a variety of configurations depending on the needs of an organization, with more or less of the effort placed on the server or the client, for example.

Security

AppsFidelity fully respects and follows the existing Notes/Domino security model. AppsFidelity works with the privileges and security level of the Domino server ID or Notes user ID it is running on.

2

Installation

This chapter will discuss exactly how to install and configure AppsFidelity to run on your Domino server. It includes examples of the most common configuration entries for your NOTES.INI file. Additional information on configuration options can be found in Chapter 3 – Configuration Options. Information on monitoring the activity of AppsFidelity, including debugging and tracing options, can be found in Chapter 4 – Monitoring and Debugging.

Preparing for Installation

In order to install and configure AppsFidelity, you will need both the software and the software license. The AppsFidelity software will be available from the Genii Software website from the AppsFidelity Download page at http://www.GeniiSoft.com/showcase.nsf/AppsFidelityDownloads, while the AppsFidelity license, whether an evaluation license or the production license, must be obtained from Genii Software or a reseller. Be sure that you have the appropriate software for your Domino server's operating system and Domino release.

1. Stop the Domino server

You cannot modify the Notes.INI information on a Domino server without first shutting down the server or stopping the Domino service. The same rule applies for upgrades to AppsFidelity.

You will need to edit your Notes.INI file to add and modify some options. Open the Notes.INI file and proceed to the next step.

2. Copy the software and license into the program directory

Copy the AppsFidelity software, named **nAppsFidelityEdit32.dll** for use with 32-bit Domino on Windows, **nAppsFidelityEdit64.dll** for use with 64-bit Domino on Windows and **libappsfidelityedit32.so** for use with 32-bit Domino on Linux, into the same directory as the Domino server software. The license file, named a**ppsfidelity.lic** for all operating systems, should be copied into the same directory as the software.

You will need to edit your Notes.INI file to add and modify some options. Open the Notes.INI file and proceed to the next step. Steps for modifying the Notes.INI information on different server platforms can be found in the appropriate Domino server documentation.

3. Modify the “extmgr_addins=” parameter

Search the Notes.INI file for the **extmgr_addins=** parameter. If it does not exist, add it at the end of the Notes.INI file, but *always make sure there is a blank line after the last entry in the Notes.INI file*. Without this blank line, the final entry may be ignored. The line added should be this on 32-bit Domino on Windows:

extmgr_addins=AppsFidelityEdit32

or on a 64-bit Domino on Windows:

extmgr_addins=AppsFidelityEdit32

or on a Linux system,

extmgr_addins=appsfidelityedit32

You may find there is already an **extmgr_addins=** parameter, as other custom add-ins also use this technology. In this case, simply add the AppsFidelity parameter to the existing parameters as a comma delimited list, so that

extmgr_addins=app1,app2

becomes

extmgr_addins=app1,app2,AppsFidelityEdit32

There can be any number of applications included in this list. While it is not required that the AppsFidelity parameter be last, it is generally recommended.

4. Add additional AppsFidelity parameters

See the Configuration Options section for a list of additional AppsFidelity parameters which can be set in the Notes.INI. The most common three that enable different aspects of the editing are below, and it is recommended that all three be set to start.

AppsFidelityClassicEnabled=1
AppsFidelityXPagesEnabled=1
AppsFidelityRecognizeDiscussion=1

5. Restart the Domino server

When you have finished making changes to the Notes.INI information on the Domino server, and made sure they are saved, you can safely restart the Domino server or Domino service. If the AppsFidelity software is properly loaded, you will see a few messages along the lines of:

```
06/09/2015 11:06:47 AM  AppsFidelity (server): Loaded into 'SERVER' process
06/09/2015 11:06:48 AM  AppsFidelity Version 3.2
06/09/2015 11:06:48 AM  Copyright (c) 2005-2015 Genii Software Ltd., All Rights Reserved
```

3

Configuration Options

This chapter will describe each standard configuration option for AppsFidelity, what values it may contain, and the impact of each value. The monitoring and debugging options are covered separately in Chapter 3.

NOTES.INI options

The only required change to the NOTES.INI file is the setting of the **extmgr_addins** line and enabling Classic or XPages and recognizing the Discussion db, all described in section 2. There are a small number of additional configuration options available.

AppsFidelityEditor=editor (CKeditor, TinyMCE, Xinha or EditLive)

This setting is not necessary when the integrated CKeditor is used, as CKeditor is assumed. If an alternate editor is to be used, setting the **AppsFidelityEditor=TinyMCE** (for example) is the equivalent of setting the **$AppsFidelityProperties** field to "AppsFidelityor=TinyMCE", but it allows you to leave that field off the form entirely. This parameter is useful if only a single rich text editor is to be used in your environment, and it is not CKeditor. There should not be quotes around the editor name.

AppsFidelityProperties=properties

Setting the **AppsFidelityProperties** value is the equivalent of setting the entire **$AppsFidelityProperties** field, which may be useful if you are using a non-standard web editor (not CKEditor, TinyMCE, Xinha or EditLive) and have set individual properties directly. Again, it is useful if you have uniform requirements on your server and want to leave the **$AppsFidelityProperties** field off the form or subform.

AppsFidelityServerTasks=tasks (HTTP or SERVER, both default to on)

This setting should not usually be set. By default, both the HTTP process and the SERVER process use AppsFidelity, which means that updates done through the web are handled by the HTTP process while updates to a server based database done from the Notes client are handled through the SERVER process. In some circumstances where performance is a consideration, it may be desired to set this to **AppsFidelityServerTasks=HTTP** so that web updates will be handled by the HTTP process, but the client based AppsFidelity process will handle all updates to both local and server based databases from the Notes client. This is not normally advisable, but may be necessary on very performance intensive or underpowered servers.

4

Implementation

This chapter will describe each standard configuration option for AppsFidelity, what values it may contain, and the impact of each value. The monitoring and debugging options are covered separately in Chapter 3.

Option 1 – Xpages (Discussion)

The most common use of editing with XPages is the standard Discussion template. In order to turn on AppsFidelity Edit support for databases based on trhe Discussion template, simply add the parameters to the NOTES.INI:

AppsFidelityXPagesEnabled=1
AppsFidelityRecognizeDiscussion=1

Once the server is restarted, AppsFidelity will handle databases using the Discussion template without any design changes.

Option 2 – XPages (Custom database defined in NOTES.INI)

For databases which use XPages but which are not based on the standard Discussion database, it is easy to enable the design for AppsFidelity use without changing the design. The parameters are listed below, but explained more completely in Appendix B:

AppsFidelityXPage*n*=template	(*n* starts with 1, template is design template name)
AppsFidelityXPage*n*=[replica_id]	(*n* starts with 1, db's replica id)
AppsFidelityXPage*n*=<filepath>	(*n* starts with 1, db's relative filepath)
AppsFidelityXPage*n*Form*m*=form	(*m* starts with 1, form with RT field)
AppsFidelityXPage*n*Fields*m*=fields	(*m* starts with 1, RT fields for form *m*)

Example (see Appendix B for explanation):

```
AppsFidelityXPagesEnabled=1
AppsFidelityXPage1=PoliciesTempl
AppsFidelityXPage1Form1=Procedures
AppsFidelityXPage1Fields1=Procedure,Justification
AppsFidelityXPage2=[85257E660050E0AE]
AppsFidelityXPage3=<cust\MegaCorp.nsf>
AppsFidelityXPage3Form1=Main
AppsFidelityXPage3Form2=Response
```

Option 3 – XPages (Custom database defined by fields)

For databases which use XPages but which are not based on the standard Discussion database, it is also possible to enable the documents for AppsFidelity use by adding computed fields to the form(s). The **$AppsFidelityXpage** field should compute to "1" (single character 1 as text value), while the The **$AppsFidelityFlds** field should compute to a text list with each rich text field listed. In addition, the **$AppsFidelityProperties** field can be used to set additional rendering properties, though this is seldom required. Additional information is available in Appendix B.

Please note: the following parameter must be set to enable any XPages processing by AppsFidelity:

AppsFidelityXPagesEnabled=1

Option 4 – Classic (Custom database defined in NOTES.INI)

For databases which use Classic design (as oppose to XPages) based on the standard Discussion database, it is also possible to enable the design for AppsFidelity use without changing the design, although there are some drawbacks. The parameters are listed below, but explained more completely in Appendix C:

AppsFidelityClassic*n*=template	(*n* starts with 1, template is design template name)
AppsFidelityClassic*n*=[replica_id]	(*n* starts with 1, db's replica id)
AppsFidelityClassic*n*=<filepath>	(*n* starts with 1, db's filepath)
AppsFidelityClassic*n*Form*m*=form	(*m* starts with 1, form with RT field)
AppsFidelityClassic*n*Fields*m*=fields	(*m* starts with 1, RT flds for form *m ; HTML flds*)

Example (see Appendix C for explanation):

AppsFidelityClassicEnabled=1
AppsFidelityClassic1=MasterProducts
AppsFidelityClassic1Form1=Products
AppsFidelityClassic1Fields1=Body,Related;BodyWeb,RelatedWeb
AppsFidelityClassic2=[85253E220050A0BE]
AppsFidelityClassic3=<SalesReports.nsf>
AppsFidelityClassic3Form1=Salesperson
AppsFidelityClassic3Fields1=Bio;BioWeb

Option 5a – Classic (Custom db defined by subforms & fields)

The most popular way to implement Classic rendering with AppsFidelity is by adding subforms that separate out the Notes client use and the web use. These

subforms can then contain the standard AppsFidelity fields defined below, as well as additional fields and events which implement the specific web editor. For example, in the Classic sample we ship with AppsFidelity 3.0, the subforms contain a computed field called HTMLBodyContent which can then be referred to in the HTMLBodyContent formula of the form itself. In addition, the JavaScript Header for the subform can be used to supply the Javascript used to launch or configure most web editors.

The subforms are usually paired so that there is one version for Notes and the other for the web for each form. In the standard **AppsFidelity Integration** database, there are four subforms used in this way. The pair **WebSubform** and **NotesSubform** are used by the **Example Form**, which only has a single rich text field. The pair **MultiWebSubform** and **MultiNotesSubform** are used by the **Multi-Field Example Form**, which has three separate rich text fields that are handled by separate editor instances. If different forms were used which use the same field names, these could be used repeatedly by the different forms (e.g., a **Body** and **BodyWeb** field might be used by a discussion database in both the **Main Topic** and **Response** forms).

Option 5b – Classic (Custom db defined by forms & fields)

The second most popular way to implement Classic rendering with AppsFidelity is by simply creating two forms that separate out the Notes client use and the web use. The forms themselves then contain the standard AppsFidelity fields defined below, as well as additional fields and events which implement the specific web editor. The best reasons for this approach is to simplify the implementation, since the HTMLBodyContent formula and the JavaScript Header can be used directly, and there is no need for additional design elements. This is often a good approach if it is only desired to use AppsFidelity with a single form and the least impact on the database is required.

5

Monitoring and Debugging

This chapter will discuss a variety of utilities that we have included for you. They will provide an easy way to get started, as well as an easy way to test your own code. Mainly, these utilities will provide easy ways to send or receive messages.

When should you use monitoring and debugging?

Only when a specific problem arises. Most of the time, AppsFidelity runs quietly in the background on your server, and there is no need to turn on any monitoring, tracing or debugging. On occasion, usually in consultation with the support people at Genii Software, you may turn on debugging to track a specific problem. Since messages from debugging accumulate quickly, only turn on debugging when necessary, and start with a AppsFidelityDebug level of 1 first. If you need additional information, turn on AppsFidelityDebug to a level of 2, but monitor the growth of the Notes Log when in this mode. When you are finished diagnosing a particular problem, remember to either remove the AppsFidelityDebug option or set it to 0. Besides the log file size, debugging can inhibit performance.

AppsFidelityDebug

Determines the debugging level. Normally, this option should be left out or set to 0, which implies no debugging. Debugging information is to the log.nsf. As debugging can inhibit performance, it should only be used when specific information is sought.

AppsFidelityDebug=0	(all debugging disabled – same as leaving option out)
AppsFidelityDebug=1	(minimal debugging enabled)
AppsFidelityDebug=2	(full debugging enabled)
AppsFidelityDebug=3	(invasive debugging enabled)

6

Troubleshooting

This chapter will discuss different potential problems and the possible causes and cures.

Editor toolbar does not appear, but textarea does

The exact cause of this may depend on the editor and implementation, but one common cause is an OnLoad JavaScript event that interferes with the window.onload code which implements the editor. If you have code in your OnLoad event, it should be moved down into the editor's subform, usually into the window.onload code in the subform's JavaScript header.

Modifications made from the web are not saved

There are a few different causes for this, but a couple of things to watch for are the **Generate HTML for all fields** property and WebQuerySave agents. The **Generate HTML for all fields** setting will cause problems with implementations that use innerHTML to get the content of the rich text field since the rich text field will show up twice on the document as a named element. Switching to an editor implementation which uses textarea replacement is recommended if this setting is necessary. If you do switch from an innerHTML to textarea implementation, do not forget to switch the rich text field on the web subform to an editable field from a computed field. You may also need to set the id of the rich text field.

IBM Discussion template integration

Integration details for the standard IBM Discussion database.

General Information

The standard IBM Discussion template can be used with no changes at all by setting the following parameter:

AppsFidelityrecognizeDiscussion=1

Turning this on enables AppsFidelity Edit to work with all databases marked as using the standard XPages Discussion database that comes with Domino 8.5.3 and later.

Licensing considerations

As there are three AppsFidelity licensing models, it should be noted that this setting does not override the licensing. If the AppsFidelity license is for one database per server, and that database is set to inherit design from the Discussion template, only that database would use AppsFidelity. If the AppsFidelity license is for one template per server, it would need to be licensed for **StdR85Discussion** or none of the databases would use AppsFidelity. The least restrictive and most common license is for the whole server, in which cases any of these databases using the Discussion template would use AppsFidelity.

B

XPages application integration

Integration details for applications built using the XPages framework.

General Information

Applications built on XPages which use rich text editing employ the built in CKEditor a little differently than Classic applications. When updated from the web, the rich text is saved as MIME rather than HTML, and in rich text when the document is saved from the Notes client. AppsFidelity follows this model, but handles the rich text to MIME and MIME to rich text conversions allowing much better fidelity and avoiding the warning messages that appear when a conversion is necessary.

Using the NOTE.INI to define templates and designs

The recommented way to add AppsFidelity to an XPages custom database is through a small set of NOTES.INI settings. Use of these settings alleviates any need to modify the design of the XPages application.

The only required statement is the identification of the design template or of a specific database by replica id or file path. The possible forms are:

AppsFidelityXPage1=template
AppsFidelityXPage1=[replica_id]
AppsFidelityXPage1=<filepath>

If **template** is used, this applies to all databases on the server with the design template set to that string. If **replica_id** or **filepath** are used, this only applies to the specific database. Multiple statements can be used for additional templates or databases, in the pattern **AppsFidelityXPage2, AppsFidelityXPage3** etc.

By default, an XPages template is assumed to have the same forms and field as the IBM Discussion template (two forms named **MainTopic** and **Response**, each with a **Body** field). If the XPages template has different forms and fields, they can be identified using AppsFidelityXPage1Form1, AppsFidelityXPage1Form2 and so forth, followed by AppsFidelityXPage1Fields1 (comma separated list of fields for AppsFidelityXPage1Form1), AppsFidelityXPage1Fields2 (fields for AppsFidelityXPage1Form2), and then followed by AppsFidelityXPage2Fields1 (fields for AppsFidelityXPage2Form1), AppsFidelityXPage2Fields2 (fields for AppsFidelityXPage2Form2) and so forth.

As an example, assume there is a design template named **PoliciesTempl** which has one form named **Procedures** with two rich text fields named **Procedure** and **Justification**, and this is used in several different databases. In addition, there is a single a single database with replica id 85257E660050E0AE which has **MainTopic** and **Response** forms, each with one **Body** field, just like the Discussion database. Finally, there is another single database with a relative filepath of **cust\MegaCorp.nsf** with two forms, Main and Response, each with a **Body** field. The NOTES.INI would then have the setting

AppsFidelityXPage1=PoliciesTempl
AppsFidelityXPage1Form1=Procedures
AppsFidelityXPage1Fields1=Procedure,Justification
AppsFidelityXPage2=[85257E660050E0AE]
AppsFidelityXPage3=<cust\MegaCorp.nsf>
AppsFidelityXPage3Form1=Main
AppsFidelityXPage3Form2=Response

Using fields to define design

There are cases where it is preferred to change the design of an application itself to indicate that AppsFidelity should handle specific forms and fields. In that case, the basic trigger to tell AppsFideliity to handle a document is the **$AppsFidelityXPage** field, which should be set to "1" (a text value of the single character, not a number). When AppsFidelity recognizes this field, it knows to treat the document as an XPage enabled document. In addition, the **$AppsFidelityFlds** value should be set to a list of the rich text fields that are editable.

Considerations of method of implementation

There are no particular performance or security differences whether you use the NOTES.INI or fields to implement AppsFidelity in an XPages template. Either approach has roughly the same impact on the server. One advantage with the NOTES.INI approach is that it can be implemented in an existing XPages application, even a design-locked 3rd party application or template. One advantage of the field approach is that in some very particular cases, the editing can be turned on or off for single documents by changing the computed value of **$AppsFidelityXPage** to "1" or "0".

Licensing considerations

As there are three AppsFidelity licensing models, it should be noted that these definition do not override the licensing. If the AppsFidelity license is for one database per server, and that replica id were one of the several databases with the shared **PoliciesTempl** design, only that database would use AppsFidelity. If the

AppsFidelity license is for one template per server, it would need to be licensed for **PoliciesTempl** or none of the databases with the design would use AppsFidelity. The least restrictive and most common license is for the whole server, in which cases any of these databases which matched the template or replica id would use AppsFidelity.

C

Classic application integration

Integration details for applications built using the classic Domino web application framework.

General Information

Applications built on Classic designs which use rich text editing can employ the built in CKEditor, though the stored results will be in HTML when editied from the web rather than in MIME as with XPages. The implication is that there are two separate fields, by default **Body** and **BodyWeb**. When the rich text **Body** is saved from the Notes client, AppsFidelity will create a synchronized HTML version in the **BodyWeb** field. When the HTML in BodyWeb is saved from a web client, AppsFidelity will synchronize the other direction to create rich text in the **Body** field.

Using the NOTE.INI to define templates and designs

Though it is recommented to add AppsFidelity to an XPages custom database with the NOTES.INI settings, there are some limitations to doing the same with Classic design, and there is still a need to adjust the web interface to use CKEditor with the second field, whether **BodyWeb** or some other name. Still, it is possible to enable a Classic application through the NOTES.INI using similar parameters to those in XPages applications.

The only required statement is the identification of the design template or of a specific database by replica id or file path. The possible forms are:

AppsFidelityClassic1=template
AppsFidelityClassic1=[replica_id]
AppsFidelityClassic1=<filepath>

If **template** is used, this applies to all databases on the server with the design template set to that string. If **replica_id** or **filepath** are used, this only applies to the specific database. Multiple statements can be used for additional templates or databases, in the pattern **AppsFidelityClassic2, AppsFidelityClassic3** etc.

By default, a Classic template is assumed to have the same forms and field as the IBM Discussion template (two forms named **MainTopic** and **Response**, each with a **Body** field). If the Classic template has different forms and fields, they can be identified using AppsFidelity Classic 1Form1, AppsFidelity Classic 1Form2 and so forth, followed by AppsFidelity Classic 1Fields1 (comma separated list of

fields for AppsFidelityXPage1Form1, followed by semicolon and the matching web fields), AppsFidelityXPage1Fields2 (fields for AppsFidelityXPage1Form2), and then followed by AppsFidelityXPage2Fields1 (fields for AppsFidelityXPage2Form1), AppsFidelityXPage2Fields2 (fields for AppsFidelityXPage2Form2) and so forth.

As an example, assume there is a design template named **PoliciesTempl** which has one form named **Procedures** with two rich text fields named **Procedure** and **Justification** (and matching web fields named **ProcWeb** and **JustWeb**), and this is used in several different databases. In addition, there is a single a single database with replica id **85257E660050E0AE** which has **MainTopic** and **Response** forms, each with one **Body** field (and matching **BodyWeb** field), just like the Discussion database. Finally, there is another single database with a relative filepath of **cust\MegaCorp.nsf** with two forms, Main and Response, each with **Body** field and **BodyWeb** field. The NOTES.INI would then have the setting:

AppsFidelityClassic1=PoliciesTempl
AppsFidelityClassic1Form1=Procedures
AppsFidelityClassic1Fields1=Procedure,Justification;ProcWeb,JustWeb
AppsFidelityClassic2=[85257E660050E0AE]
AppsFidelityClassic3=<cust\MegaCorp.nsf>
AppsFidelityClassic3Form1=Main
AppsFidelityClassic3Form2=Response

Using fields to define design

There are cases where it is preferred to change the design of an application itself to indicate that AppsFidelity should handle specific forms and fields. In that case, the basic trigger to tell AppsFideliity to handle a document is the **$AppsFidelity** field, which should be computed to the value of @ClientType (i.e., “Notes” when opened in Notes client and “Web” when opened in web client). When AppsFidelity recognizes this field, it knows to treat the document and whether the direction of synchronization is from rich text to HTML or HTML to rich text.

In addition, there are two fields which determine the rich text fields and HTML fields. The **$AppsFidelityFlds** value should be set to a list of the rich text fields that are editable. The **$AppsFidelityFldsH** value should be set to a list of the corresponding HTML fields to be used on the web.

Standard AppsFidelity fields

The **$AppsFidelity** field should always compute to “Web” when the document is saved on the web, and to “Notes” when the document is saved from the Notes client. In Notes 6 and above, the formula can be set to @ClientType, but in R5 it

would need to check the roles to determine whether “Web” or “Notes” should be computed.

The **$AppsFidelityProperties** field is set to “AppsFidelityor=’CKEditor’ ” or “AppsFidelityor=’TinyMCE’ ” or “AppsFidelityor=’Xinha’ ”, but the field is usually left out completely if the CKEditor which comes with the Domino server is to be used. There are additional properties, but they should only be set under consultation with Genii Software support. The default editor is CKEditor if none of these options are available.

The **$AppsFidelityFlds** field is a multi-value field set to the names of the rich text fields which should be editable with the Notes client but which have matching web rich text fields in the **$AppsFidelityFldsH** field. If this field does not exist, it will default to a value of “Body”.

The **$AppsFidelityFldsH** field is a multi-value field set to the names of the rich text fields which should be editable with the CKEditor and which have matching Notes rich text fields in the **$AppsFidelityFlds** field. If this field does not exist, it will default to a value of “BodyWeb”.

Additional Fields used in samples

The **HTMLBodyContent** field is computed to include the CKEditor references to style sheets and other resources used by the editor. It is not used directly by AppsFidelity, but is simply a convenient way of storing the content used by the HTML Body Content formula on the main form.

Considerations of method of implementation

There are few performance or security differences whether you use the NOTES.INI or fields to implement AppsFidelity in a Classic design, but there are cases where the direction of synchronization is not obvious, especially from an agent changing the Body field from the web, for example. In addition, since there are almost always design changes needed to implement the CKEditor in the Classic design, it usually makes more sense to use either the subforms used in the downloadable AppsFidelity Integration databasr, or forms used for the same purpose.

Licensing considerations

As there are three AppsFidelity licensing models, it should be noted that these definition do not override the licensing. If the AppsFidelity license is for one database per server, and that replica id were one of the several databases with the shared **PoliciesTempl** design, only that database would use AppsFidelity. If the AppsFidelity license is for one template per server, it would need to be licensed

for **PoliciesTempl** or none of the databases with the design would use AppsFidelity. The least restrictive and most common license is for the whole server, in which cases any of these databases which matched the template or replica id would use AppsFidelity.