An Information Technology Perspective
SharePoint and Legacy Screen-Based Applications

This document provides a technical perspective on goals, challenges and solutions in improving business processes through the use of SharePoint and the Flynet Viewer product family.

In this document, SharePoint refers to WSS 3.0, MOSS 2007 and SharePoint 2010 versions. The Flynet Viewer products integrate with legacy screen-based applications, including IBM Mainframe, AS400/iSeries and DEC/VAX/Unix applications typically accessed with PC-based terminal emulators.

Assumed Goals

The primary goal is assumed to be the use of SharePoint as a focal point for delivery of improved productivity to business departments by an Information Technology organization or a vendor tasked with providing I.T. consulting/services.

A secondary goal, but core to this document, is integrating into the SharePoint deployments legacy screen-based applications that the business departments currently directly or indirectly utilize in performing their existing, pre-SharePoint business processes.

Relating to the integration of legacy screen-based applications, it is assumed that the full migration or modernization of the screen-based applications is not desirable or cost effective as it relates to a SharePoint implementation. However, any investment in product or services relating to the legacy applications should not act as a lock-in or act as a technical deterrent to future modernization or migration efforts.

Assuming a successful implementation of well designed navigation, content, work flow management and other common SharePoint deliverables by the I.T. project team, the delivered environment should provide a platform for ongoing improvement. This includes both I.T. provided enhancements (such as custom business process management implementations) as well as user-department enhancements utilizing the high-level collaboration and content-management capabilities of SharePoint that include search, ad-hoc lists, lightweight workflows and custom web part pages that link-together existing components to address user requirements on an as-needed basis.

Challenges

While there are many challenges involved in achieving a highly-functioning SharePoint site including business-side cooperation, business process identification, design and execution, this document will focus on the challenges involved with legacy screen-based applications.

The principal challenge is: what to do with these old, screen-based applications? If a business unit is to have its work upgraded with a web-based product such as SharePoint, users currently accessing legacy applications with traditional fat-client terminal emulators may not fit well in the design, both technically and organizationally. The cryptic, function-laden user interfaces of typical legacy screen applications can force users into specialized roles due to high training requirements. This then can be a disrupting factor when a business process upgrade moves functions across the unit to different users (or new hardware such as a mail-room scanner).

If users currently send forms or other media to a back-office operation where screen-based applications are utilized, this too can present a challenge to a optimized SharePoint implementation when the back-office workers are operating in a closed terminal emulation environment.

Often, users are not principally utilizing terminal emulation for all of their work but may need to use screen applications for certain data entry, customer service or decision support activities. Whatever the work may be, if it is being performed in a terminal emulator it can be a roadblock for a clean SharePoint integration.

Solutions using Flynet Viewer Tools & Middleware

Legacy screen-based applications that are currently locked into terminal emulators can be integrated with a SharePoint solution using a number of Flynet features, including:

- SharePoint-compatible terminal emulation (with Scripting)
- SharePoint list web part integration
- SharePoint forms data entry using screen data
- Screen functional access using Enhanced ASP.NET Web Pages
- Screen functional encapsulation using web services

Each of these is described in the following sections.

SharePoint-Compatible Terminal Emulation (with Scripting)

The Flynet Host Access Web Part for SharePoint provides versatile runtime management of user screen sessions, including highly compatible terminal emulation when needed.

This is delivered using the Flynet Web Terminal emulator, which is a Web 2.0, highly functional terminal emulator with auto-sized fonts, full keyboard mapping, keyboard buffering and other features normally associated with traditional desktop terminal emulators. Despite all this traditional functionality, the Flynet Terminal Emulator is a SharePoint-compatible, pure HTML and CSS implementation that can be tailored using custom graphics and CSS file enhancements to maintain consistency with the look and feel of the SharePoint environment.

Along with compatibility with SharePoint, the Flynet Host Access web part can deliver integration with SharePoint data using a high-level scripting capability that can provide 2-way data integration between SharePoint and the screens, including host navigation and screen-based data entry. Examples of this data integration are provided in the following sections.

The minimum requirements for scripting each host application area include recording screens with navigation and any relevant data entry, identifying screen names using Flynet Viewer Studio as well as mapping those fields needed for application support. The rest is handled by the Flynet application generation, which creates a custom integration framework along with the SharePoint Flynet Host Access Web Part. With the requirements taken care of, a custom SharePoint-integrated script editor provides high-level definition of how the screens should interact with other web parts and forms on a particular SharePoint document.

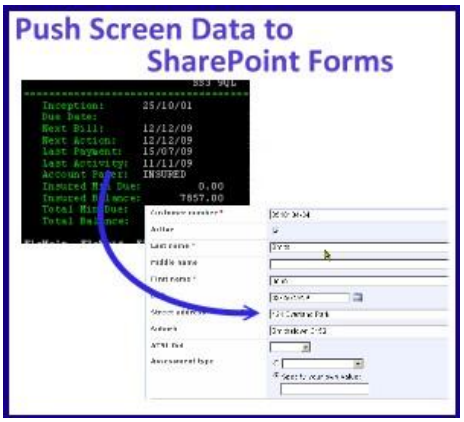
SharePoint List Web Part Integration

Using the high-level scripting feature of the Flynet Host Access Web Part, SharePoint list items can be connected to screen actions. This works for the Flynet Terminal Emulator as well as for the Enhanced ASP.NET Web Pages described in a later section of this document.

For example, a new web part document can be created in SharePoint containing a list of purchase orders. By adding the Flynet web part to the same document, a row connection can be set linking the list of purchase orders to the Flynet web part. With a short script property set in the Flynet Part, each click by a user on a list item can use the contents of the list to navigate to a selected screen in the host as well as enter other fields from the list row into the screen.

While this example describes a row connection driving screen interaction from a SharePoint list, note that any web part that can provide a row connection can do the same, as the Flynet Host Access web part provides a generic row consumer that can match and convert any field contained in any SharePoint row for use in a screen integration script.

SharePoint Forms Data Entry Using Screen Data



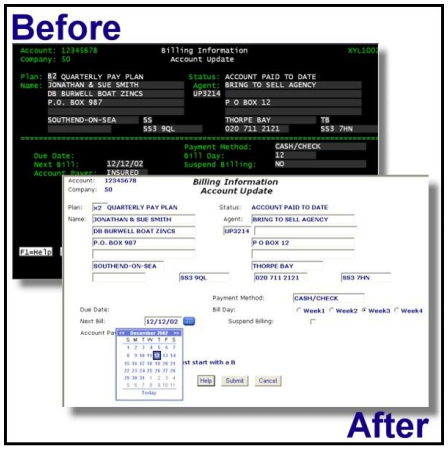
As departments create new lists and lightweight workflows in SharePoint, users may find themselves creating new list items using host application data as a source. With the Flynet Host Access Web Part, a special client-side Javascript capability can match screen-based fields with the fields in a SharePoint New Item, enabling the auto-entry of selected fields with data from one or more host screens.

In this mode, the script detects the presence of a specific screen and displays a button in the Flynet Web Part that when clicked, copies the Flynet-sourced field contents to the matching fields in the SharePoint form, ready for the user to complete the New Item form and submit it.

In addition to the ability to auto-enter data sourced by Flynet, the script can also add a button to the SharePoint form for reading SharePoint-sourced data and sending that to a custom method contained in the Flynet application (similar to a web service call).

As with the SharePoint row connection example using lists, SharePoint forms can be auto-entered using data from both the terminal emulator screens as well as the enhanced web pages described in the following section.

Screen Functional Access using Enhanced ASP.NET Web Pages



So far, examples of integration with SharePoint have used the simple terminal emulator, which may be a quick way to integrate for users that are already using a terminal emulator today. But for users unaccustomed to the screen-based applications, or for users looking for an improved and more productive user experience, Flynet provides the ability to convert and upgrade selected screens to a set of coordinated web pages.

Upgrading screens is important for many business processes being addressed in a SharePoint site, taking the functionality available in the Flynet Host Access Web Part and enhancing it beyond the raw presentation of a terminal emulator. In this mode, the same screen session that may be available for terminal emulation provides the backing data access layer for all-new ASP.NET web pages.

This made possible by features in Flynet Viewer Studio that build on the same work done for the scripting of Terminal Emulation (recording, identifying and field-mapping screens). In the Studio, a business analyst can upgrade entry fields in-place to web controls that require less application knowledge and training. For example, a field can be upgraded from a simple data entry field to a selection list that contains only those codes valid for data entry.

With the individual fields upgraded, the analyst can then convert single host screens to one or more web pages, or combine selected fields from multiple screens into a single web page. Along with the one-to-many and many-to-one capabilities, the Flynet Studio also supports popular page compositions including tabbed notebook, simple workflow and data entry wizard.

After designing the web pages, the application is generated again, producing the Flynet Host Access web part along with new enhanced ASP.NET web pages. At this point, a Visual Studio developer, using the best of that tool's design features, can further enhance the pages, since they are generated as standard ASP.NET ASPX pages with simple code-behind files.

The Visual Studio developer can test the functional area involving one or more screens using standard Visual Studio debugging, without the need for deploying to SharePoint; all the Flynet host access features that run alongside SharePoint in the production environment are also available on the developer's installation.

Once the functionality as a web user interface is ready, the application is deployed to the SharePoint server along with an updated version of the web part which includes access to all new screens, fields and available navigation.

As with the terminal emulator, the new, enhanced user interface for the upgraded screens can be accessed and scripted with the Flynet Host Access web part. By integrating with a specific set of SharePoint documents, lists or workflows, the enhanced user interface can avoid duplicate data entry while delivering ease-of-use and better productivity than the screens providing the actual data access and validated data entry.

Screen Functional Encapsulation Using Web Services

Sometimes, a user interface to a set of screens is not what is needed to support new SharePoint-delivered business value. What is needed instead is a high-level ability to utilize the functionality buried behind those screens. Examples include inquiries that leverage complex logic or custom data access not reproducible using direct access to a backend database, or data entry that must be validated and processed using existing business rules that are entwined in the screen-based transaction logic.

For these challenges, the same Flynet screen recording, naming and mapping performed for the scripting and user interface enhancements can be leveraged to create standard web services. Using Flynet Viewer Studio's Web Service modeling wizards, screen workflows are identified, properties set and a standard ASP.NET Visual Studio Web Service solution is generated.

As with the enhanced user interface, the generated solution can be loaded, edited and debugged using Visual Studio, before deploying. As part of the Flynet Studio support, there is even a Windows Form-based test harness option. This creates a customized graphical user interface application that automates testing of an individual web service, including the ability to define, save and restore the input parameters for specific test cases.

Designed to handle the most demanding and complex inquiry and data entry tasks, the Flynet Web Service capability can reliably encapsulate screen-based functionality and publish it with a fraction of the development work required from alternative approaches.

Once encapsulated as a web service, the screen-based functions can then be made available for integration with a variety of SharePoint activities. This includes InfoPath forms data entry, workflow actions and any number of custom solutions, including business process management packages that support and leverage web services.

Conclusion

For those in I.T. who are supporting SharePoint Site implementations that would benefit from including legacy screen-based applications instead of ignoring them (and hoping they will just go away!), the Flynet tools and SharePoint runtime can provide a high value solution to an otherwise unsolvable problem.

By leveraging Flynet and enabling the inclusion of the screen-based applications in the analysis and design of SharePoint solutions, those solutions will have a much higher value to the business departments they are deployed for.