SOP Reference #: SOP-IT006

Operation/Task:	Software Development			Equipment:	NA
Owner:	Vice President of Technology	Date Created: Revision History:	5/12/15 See last page	Department:	All IT Staff

ALERTS (see below): Critical Step ◆ Quality Check ☑ Tip ☺ Team Safety ✚

Purpose: This SOP/work instruction documents procedures for ENPOINTE Software Development Standards.

Step #	**Alerts**	**Step Description - "What to Do"**	**"How to Do it"**	**"Why to Do it"**
1	◆	**Introduction** The purpose of this document is to specify tools, procedures and conventions to try to provide • Adequate standards for all stages of Application Development. • Minimum requirements for Application Development activities. • A general measurement that the application development methodology is in compliance with the generally accepted standards.	**Scope** • Pageflex – Storefront Development • Middleware and Scheduled Processes development • All Externally Facing Websites • All Internal Websites • Pageflex Template Development Report Development	
2		**Development Environment** No software development is done on Production. All software development is done on developers desktop and the following development servers.	**Development Servers** **Web/App Server (Development)** • BP-TestPF08 – Development/Test Server for websites and Pageflex templates • BP-Prodexsvr04 – Development/Test Server for Middleware Components **Database Server (Development)** BP-SQLTesting01 – Development Database Server	
3		**Programming Languages** • C#, VB.Net, JavaScript, TSQL	**Software Versions** • The following are the core software tools used for development • Microsoft.Net Framework • Visual Studio 2022 • Microsoft SQL Server 2019 • SQL Server Management Studio 18 • Pageflex Server 9.9 • Pageflex Studio 9.9	Consistent language is important in producing easy to read and maintain code.

4		**Website/Application Structure** All websites will use a standard folder/directory structure.	**Pageflex-Storefront Websites** Folder Structure for Storefront Websites is defined by the Standard Storefront implementation. Any custom code must be placed in the following location within the Storefront Website Folder **HTML and JavaScript** – webpages\custom and webpages\angular **Storefront Extension DLLs** – bin\services **Custom dlls** – bin **Custom aspx pages** – webpages (website root folder) **Non-Storefront Websites** • Websites must distinguish the contents such as HTML, Server-side code, images, stylesheets, • JavaScript etc. in separate folders. Websites that are not built on Pageflex-Storefront must use the following structure at the minimum. Any additional folders can be used as required by the design. • Website Root ○ |_Images ○ |_CSS ○ |_Scripts	Using a standard folder structure would allow developers familiar with one program to easily adapt to another program following a similar standard.

5		**General Naming Conventions** All code must follow the proper naming conventions.	Do not use words that will conflict with Keywords Choose words that are easily readable Choose words that would identify the context of the contents **Naming Files** • Source Code files must represent the function of the code contained • Do not use Spaces in file names **Application Code** • Avoid using underscores, hyphens, or any other non-alphanumeric characters • Avoid using Hungarian Notations (prefixing with data type identifier) • Avoid using abbreviations or acronyms as parts of identifier names. • Use abbreviations or acronyms only if they are widely accepted. • Use Pascal Casing for public member, type, and namespace names consisting of multiple words. The first letter in the identifier and the first letter of each subsequent concatenated word are capitalized. You can use Pascal case for identifiers of three or more characters. Example: FirstName, ItemUom, UserID • Use Camel Casing for parameters and local variables. The first letter of an identifier is lowercase and the first letter of each subsequent concatenated word is capitalized. Example: firstName, itemUom, userID **Indentation** Indentation should be used to emphasize the body of control statements, loop statements, classes and methods and all scope blocks	Proper and consistent naming is important in producing easy to read and maintain code.

6		**Application Configuration Conventions** All configuration settings must be set/stored in a standardized way.	Store all application or website configuration variables in a file or location that leads to quick and easy updating. This will allow for rapid responses to requests for environmental changes. Recommended locations for configuration variables: • App.config files • Web.config files • The Configuration table in your application's database, if it has one. Store the connection string to the database in the app or web .config file. Examples of data that should be stored in the locations above: • Service Account or User login information • E-mail addresses • File paths, locations or names • Scheduling and/or retry count information Do not store application or website configuration variables within compiled code.	Proper and consistent configuration settings is important in producing easy maintain application settings.
7		**Database Scripts** All SQL scripts, views and stored procedures must follow the proper naming conventions..	• Do not use spaces or hyphens in variable names and database object names • Use upper case for all T-SQL constructs • Use lower case for all datatypes • Use PascalCasing for user defined objects and variables • Use the following for naming database objects usp_<Store Procedure Name> - User Stored Procedures usf_<Scalar Valued Function> - Scalar Valued Function utf_<Table Valued Function name> - Table Valued Function tr_<Index Name> - Indexes v_<View Name>- Views FK_<Foreign Key Name> - Foreign keys DF_<Default Name> - Defaults IX_<Index Name> - Indexes	Proper and consistent scripting is important in producing easy to read and maintain SQL.

8		**Comments** All comments must follow the proper formatting conventions.	Comments must be placed before the code that is commented for and briefly explain the purpose of the code. Avoid using flower-box comments in Visual Studio projects because they are not supported by XML comments. Use XML comments instead (// and /// for c# and ‘’ and ‘’’ for vb.net) Example- avoid using the following format /** * this is a comment **/ Flower box comments are OK in JavaScript and TSQL **File Comments** Always add comments when Checking-In and Checking-Out of GitHub All Source Code Files must have comments on the top of the file and provide the following information. **a) Change Date** **b) developer name** **c) brief description** **Example** **C#** /// 04/01/2011 Venki Updated from .net 2.0 /// <summary> /// Typically run at end of every month. /// This job will load receipts from Printstream /// Will add any new product from Printstream to the reporting database /// </summary> /// <remarks></remarks> The change history must be recorded every time the file is modified.	Comments are critical to understand source code. All scripts C#, VB, JavaScript, TSQL and CSS must use comments to describe the function of the code. **Inline Comments** Use inline to comment all of the major code blocks of the code and the critical minor points that can be easily overlooked. **Method and Property comments** All c# and vb.net Functions and Properties must be marked with **xml comments** (///) before the beginning of the Function or Property. **JavaScript Comments** Avoid using elaborate comments in JavaScript. JavaScript is rendered in browsers as is so comments are exposed to users and also can add to the browser load. (Note: This restriction can be overcome when JavaScript whitespace removal is implemented)

9		**Error Handling and Logging** All error handling and logging must follow proper standard conventions.	Error Handling and Logging functions for Storefront websites are provided by Standard Storefront methods. • All custom code on the storefront website must **try…catch…throw** or **Storefront.LogMessage** function to log messages to the storefront database. • Do not use try/catch blocks for flow-control. • Always catch exceptions that can be handled in the order of most derived to least derived exception. • Only catch predictable exceptions. • Never declare an empty catch block. • Avoid nesting a try/catch within a catch block. • Avoid re-throwing an exception. Allow it to bubble-up instead. If re-throwing, always use the stack trace	Proper and consistent error handling is important in producing easy to read programs for debugging.
10		**Software Documentation** All software documentation must be located in a common place and follow standard identifiers	• All Software Development related (Technical) documentation is stored in the folder **\\bp-file01\Groups\DIT\IT_Documentation_Library** • All Technical Documents must have the following items **Title** **Created By** **Table of Contents** **Change History** **Introduction**	Proper and consistent documentation is important in producing easy to find and read and maintain applications and services.

11		**Using Source Control (GitHub)** All code, templates and scripts must use Source Control..	• To modify existing code, always use the latest version from GitHub. Any change to existing code/template is only done on the code saved in GitHub. • Any new code must be **checked into** GitHub before promoting to Production. • A copy of all the code and Pageflex Templates is saved in GitHub. • Always check-in before migrating code to production. • Always add comments during Checkout and Check-in files on GitHub. • Do not Check-in file that is unfinished or not tested. **Visual Studio Projects** Visual studio has built in support for Source Control using GitHub. • All production solutions/projects must be added to GitHub once they are created. • Only work on code from GITHUB for any changes that would affect production. • Always Undo Checkout if changes are made temporarily for testing. **Pageflex Projects** Pageflex Studio does not have built in support for GITHUB • Always get latest version from GITHUB before modifying a Pageflex Project for production. • Keep files checked out during work in progress • Always manually add or update project to GITHUB using Team Explorer before migrating code to production.	Proper and consistent use of source control is important in producing easy to find and maintain code, templates and scripts.
13		**Using Source Control (GitHub) (continued)**	**Database Objects/SSIS 2018** SQL Server Management Studio does not have built in support for GITHUB. Visual Studio does not use built in support for GITHUB. • Always get latest version from GITHUB before modifying Database Scripts and SSIS Packages. • Always Keep files checked out during work in progress • Always manually add or update project to GITHUB before migrating code to production. • Do not add scripts with DROP statements to GITHUB.	

14		**Deployment of code to production** All production code deployments must follow the standard deployment instructions.	• Migrating changes to production is initiated via Track-It Change Management. • The standard form **Application Migration - Instructions Template.doc** located at \\bp-file01\Groups\DIT\IT_Documentation_Library must accompany all code migration requests.	Proper and consistent deployment documentation and procedures is important in producing easy to deploy applications and services.

Notes:

Definitions:

Revision History	**Description of Changes**	**Requested by**	**Date**
Rev 1	Converted to new SOP template layout	Steve Kirk	5/7/15
Rev 2	Added Revision History table	Steve Kirk	8//16
Rev 3	Revised How to Do It in Steps 9, 12, and 13; changed gray header date information	Dave Johnson	3/31/20
Rev 4	Changed GLS reference to ENPOINTE in Purpose	Cristi Oakvik	3/9/21
Rev 5	Changed Owner to Director of Business and custom Applications. Updated version numbers throughout. Removed #8 Pageflex Studio Templates.	Mike Starrett	4/28/23
Rev 6	Replaced owner with Vice President of Technology. Replaced TFS with GitHub. Updated software version numbers.	Mike Starrett	11/4/24